California State University Fullerton

CPSC 462



Object Oriented Software Design

Design Model

for the



Hotel Reservation

System

|  |  |  |
| --- | --- | --- |
|  | **Josh Ibad** |  |
|  | Chief Software Architect |  |
|  | [joshcibad@csu.fullerton.edu](mailto:joshcibad@csu.fullerton.edu?subject=Regarding the HotelBuch System) |  |

Revision History:

| Version | Date | Summary of Changes | Author |
| --- | --- | --- | --- |
| 1.0 | 2021-11-15 | * Initial Release | Josh Ibad |

Table of Contents

[1 Static View 1](#_Toc50298153)

[1.1 <Story Title 1> 1](#_Toc50298154)

[1.1.1 Software Class Diagram 1](#_Toc50298155)

[1.1.2 Description 1](#_Toc50298156)

[1.2 <Story Title 2> 1](#_Toc50298157)

[1.2.1 Software Class Diagram 1](#_Toc50298158)

[1.2.2 Description 1](#_Toc50298159)

[1.3 <Story Title 3> 1](#_Toc50298160)

[1.3.1 Software Class Diagram 1](#_Toc50298161)

[1.3.2 Description 1](#_Toc50298162)

[1.4 <Story Title X …> 2](#_Toc50298163)

[1.4.1 Software Class Diagram 2](#_Toc50298164)

[1.4.2 Description 2](#_Toc50298165)

[2 Dynamic View 3](#_Toc50298166)

[2.1 <Message Name 1> Sequence of Execution 3](#_Toc50298167)

[2.1.1 Software Interaction Diagram 3](#_Toc50298168)

[2.1.2 Description 3](#_Toc50298169)

[2.1.3 SSD Traceability 3](#_Toc50298170)

[2.2 <Message Name 2> Sequence of Execution 3](#_Toc50298171)

[2.2.1 Software Interaction Diagram 3](#_Toc50298172)

[2.2.2 Description 3](#_Toc50298173)

[2.2.3 SSD Traceability 4](#_Toc50298174)

[2.3 <Message Name 3> Sequence of Execution 4](#_Toc50298175)

[2.3.1 Software Interaction Diagram 4](#_Toc50298176)

[2.3.2 Description 4](#_Toc50298177)

[2.3.3 SSD Traceability 4](#_Toc50298178)

[2.4 <Message Name X …> Sequence of Execution 4](#_Toc50298179)

[2.4.1 Software Interaction Diagram 4](#_Toc50298180)

[2.4.2 Description 4](#_Toc50298181)

[2.4.3 SSD Traceability 4](#_Toc50298182)

# Static View

NOTE TO STUDENTS:

1. See Larman §1.5 A Short Example, §14.4 Design Objects: What are Static and Dynamic Modeling, Chapter 16 in entirety
2. Software Class Diagrams. Make each diagram a new, numbered subparagraph. Describe what you want the reader to take away after reading this diagram (what story is this diagram supporting)
3. There is no prescribed number of diagrams, but you should make sure your interfaces, hierarchies, domain concepts, derived software concepts, and (important!) relationships with cardinality, composition, association, visibility, and navigability are shown. Remember lines between classes have an arrow on one end (vice lines on a Domain Model which have no arrows). An arrow on both ends is usually an errored attempted to represent a many-to-many relationship.
4. All interfaces and all design decision snippets listed in the SAD should be a small portion of this big picture.
5. All Objects on your Sequence Diagrams should be realized from a class in this static view of the software design
6. Delete this NOTE before you deliver

## <Story Title 1>

### Software Class Diagram

<insert your class diagram here>

### Description

<Describe your diagram as you tell your story in words here>

## <Story Title X …>

### Software Class Diagram

<insert your class diagram here>

### Description

<Describe your diagram as you tell your story in words here>

# Dynamic View

NOTE TO STUDENTS:

1. See Larman Chapter 15, UML Interaction Diagrams
2. Software Sequence Diagrams. Make each diagram a new, numbered subparagraph. Describe what you want the reader to take away after reading this diagram (what story is this diagram supporting)
3. There is a strong, direct relationship between these Software Sequence Diagrams (SwSD) and your System Sequence Diagrams (SSDs). Be sure to maintain this relationship.
4. Software Sequence Diagrams (SwSD) ALWAYS start with an anonymous message received at a Domain Layer Interface, and ALWAYS end with an anonymous response. Your goal is to trace the flow of execution through the Doman Layer in great, laborious detail. Leave no stone unturned. Your sequences should show a Domain Layer Object consuming the Technical Services Layer’s interfaces, but do not go deeper into the Technical Services Layer.
5. Every message from the Actor to the System in every SSD is a sequence to be diagramed here. Not every SSD, but rather every message on the SSD. In other words, you need to provide a sequence of execution through the software for every request made by an Actor. An SSD may produce 10 or 12 Software Sequence Diagrams. Do not repeat sequences. For example, if two SSDs show the same “Generate Weekly Consumption Report” request, and the system responds in the same way with the same information, then only one Software Sequence Diagram is required.
6. Every Scenario is a single path through the software – no decisions are depicted, and all loops have specific iteration counts and terminating criteria. Advanced diagraming provides tools to combine scenario sequences, but we are not yet advanced modelers. Sequence Diagrams have no decisions, and all loops shall be explicitly enumerated.
7. All Objects on your Sequence Diagrams should be realized from a class in this Static View of the software design
8. Delete this NOTE before you deliver

## <Message Name 1> Sequence of Execution

### Software Interaction Diagram

<insert your sequence (typical) or collaboration diagram here>

### Description

<Describe your diagram as you tell your story in words here>

### SSD Traceability

<Identify the source SSD(s) for this message (aka request). (Remember SSDs are part of the Use Case Model; a requirements document.) Cross check that the message is the name of a function in a Domain Layer Interface, and that Domain Layer Interface is part of the above Static View (and the SAD’s Logical View)>

## <Message Name 2> Sequence of Execution

### Software Interaction Diagram

<insert your sequence (typical) or collaboration diagram here>

### Description

<Describe your diagram as you tell your story in words here>

### SSD Traceability

<Identify the source SSD(s) for this message (aka request). (Remember SSDs are part of the Use Case Model; a requirements document.) Cross check that the message is the name of a function in a Domain Layer Interface, and that Domain Layer Interface is part of the above Static View (and the SAD’s Logical View)>

## <Message Name 3> Sequence of Execution

### Software Interaction Diagram

<insert your sequence (typical) or collaboration diagram here>

### Description

<Describe your diagram as you tell your story in words here>

### SSD Traceability

<Identify the source SSD(s) for this message (aka request). (Remember SSDs are part of the Use Case Model; a requirements document.) Cross check that the message is the name of a function in a Domain Layer Interface, and that Domain Layer Interface is part of the above Static View (and the SAD’s Logical View)>

## <Message Name X …> Sequence of Execution

### Software Interaction Diagram

<insert your sequence (typical) or collaboration diagram here>

### Description

<Describe your diagram as you tell your story in words here>

### SSD Traceability

<Identify the source SSD(s) for this message (aka request). (Remember SSDs are part of the Use Case Model; a requirements document.) Cross check that the message is the name of a function in a Domain Layer Interface, and that Domain Layer Interface is part of the above Static View (and the SAD’s Logical View)>