California State University Fullerton

CPSC 462

Object Oriented Software Design SW Architecture Document (SAD) for the Hotel Reservation System

Josh Ibad

Chief Software Architect

joshcibad@csu.fullerton.edu

Revision History:

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

Table of Contents

1 Architectural Representation

After careful deliberation, I have determined that it is best to follow a Session Controller GRASP pattern when it comes to the Architecture Decision of the Controller GRASP Decision. The Session Controller GRASP pattern seems more appropriate for the Hotel Reservation system considering the granularity of the system's design and all the components in the Domain layer (to be discussed shortly). The alternative, the Facade Controller pattern, would have provided a singular system-level interface at the Domain level, that would have had high dependencies on all components of the Domain level. This means that the Facade option would have introduced high coupling in the system which would have made it harder to maintain and evolve. Namely, it scales horribly in the face of a growning number of functionality and classes within the components.

In comparison, the option of choice, the Session Controller pattern, is a use-case level interface at the Domain level, with one interface for every one or many use cases. Approximately one session controller interface would exist for each component, allowing for high cohesion amongst the classes or subcomponents within the component or subpackage. This also minimizes the coupling amongst components, which allows for highly modularized and highly scalable designs, that scales well when more functions and classes are added into the system in subsequent iterations. Thus, the Session Controller option is the option of choice when it comes to the Controller GRASP decision.

With that in mind, the overall packaging and hierarchy of the system is to be highly organized in a multi-tier / layer architecture so as to separate concerns or responsibilites amongst layers. Namely, there should be a UI layer responsible for receiving user input and formatting output, as well as relaying events into the lower layers of the system. A Domain layer in turn, would be responsible for business logic. The Domain layer will highly reflect the concepts of the Business domain and the relationships and functionalities between them. These two higher layers then relay inforamtion to the TechnicalServices layer, which is responsible with low-level responsibilities such as Secondary Storage management / persistence as well as the logging of events.

In further granularity, the Domain layer should have three components: Hotel, Reservation, and Session, each having a single interface that serves as a handler (as discussed above with the Session Controller decision). The Hotel component will be responsible for managing the Hotel as a container of Rooms, along with information relating to Rooms. The Reseration component in turn, will be responsible for managing Reservations, and more specifically, the reservation times and payment information for the reservation. The Reservation's association to Rooms will be minimized to produce low coupling between the two components. Lastly, the Domain layer will also have a Session component, which is responsible for managing user roles and permission. The Session does not lead directly to the two components, but rather, will direct the UI as to when it should use the other component's functionalities, and only exposes the functionalities a certain user's Session is allowed to have.

The TechnicalServices layer will have the typical components of Persistence and Logging. The Persistence component will, down the line, manage the persistence of data through CRUD operations, freeing all other components of the responsbility of writing and reading to files or databases. The Logging component in turn will manage the logging of system events in a well formatted manner, freeing all other components of the responsibility of excessive formatting.