2 Architectural Decisions

2.1 Controller GRASP Decision

2.1.1 Decision to be made

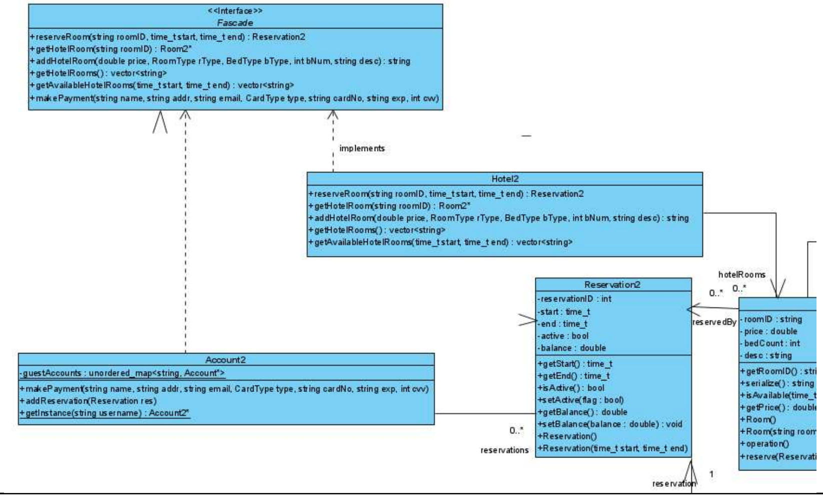
Who should be responsible for handling an input system event whenever a user attempts to manage hotel rooms?

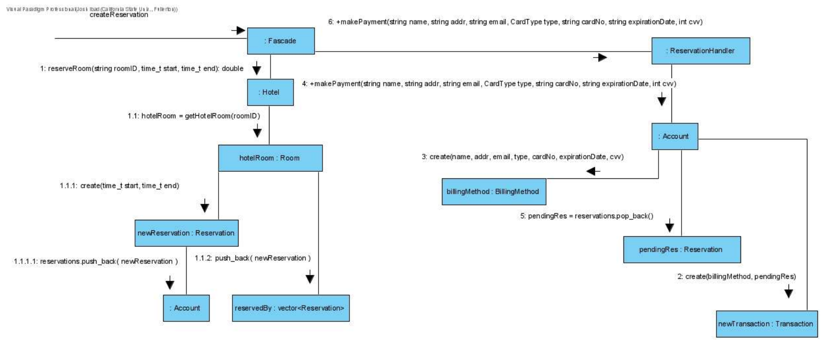
A good decision should provide good abstraction and minimize coupling by making sure that system level events are only concerened with system level details and have implementation details hidden below.

A poor decision would have too many Classes or interfaces that must be interfaced with for a simple procedure, and would be unscalable as classes and interfaces are added when the system grows.

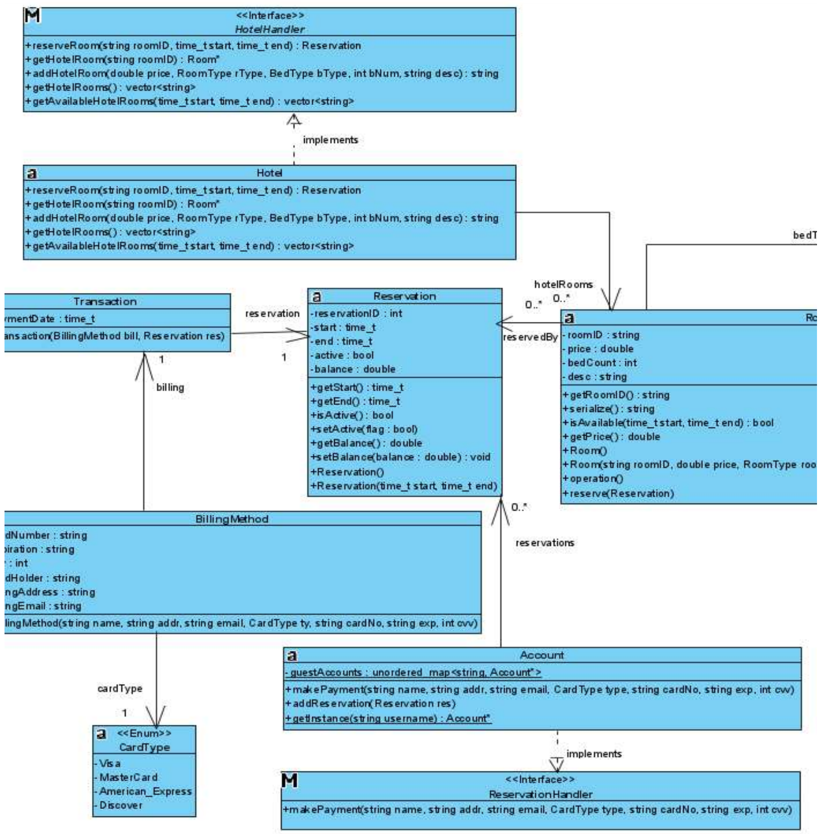
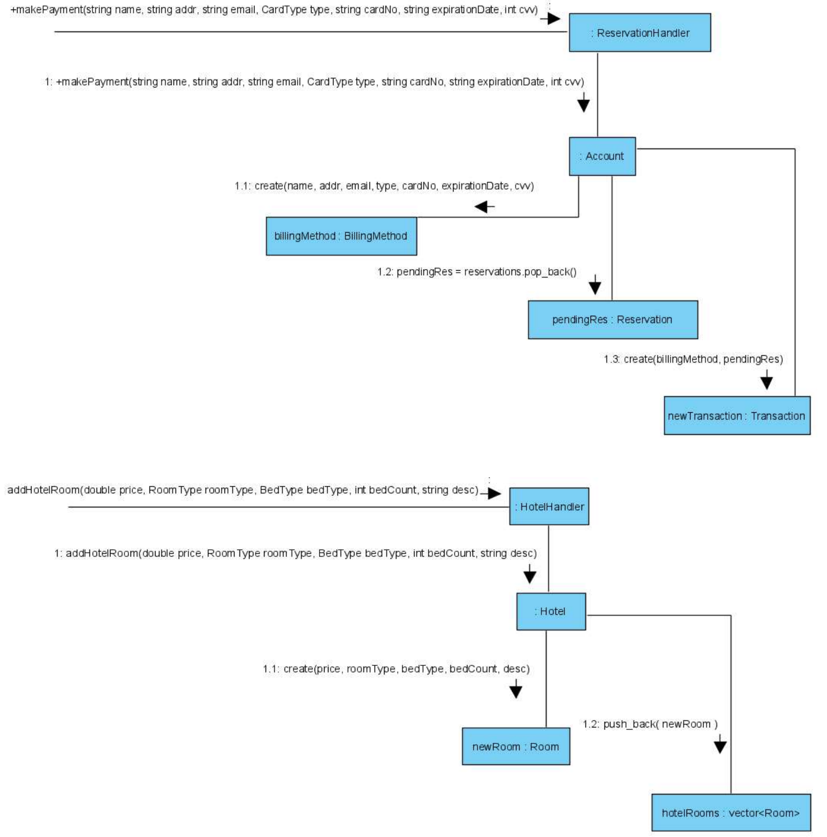
2.1.2 Options Considered

<table>
<tr><th>Controller</th><th>Static View</th><th>Dynamic View</th></tr>
<tr><td>Option 1: Facade (System) Controller

(Rejected)</td><td>
<<Interface>>
Fascade
+reserveRoom(string roomID, time_t start, time_t end) : Reservation2
+getHotelRoom(string roomID) : Room2*
+addHotelRoom(double price, RoomType rType, BedType bType, int bNum, string desc) : string
+getHotelRooms() : vector<string>
+getAvailableHotelRooms(time_t start, time_t end) : vector<string>
+makePayment(string name, string addr, string email, CardType type, string cardNo, string exp, int cvv)
implements
Hotel2
+reserveRoom(string roomID, time_t start, time_t end) : Reservation2
+getHotelRoom(string roomID) : Room2*
+addHotelRoom(double price, RoomType rType, BedType bType, int bNum, string desc) : string
+getHotelRooms() : vector<string>
+getAvailableHotelRooms(time_t start, time_t end) : vector<string>
hotelRooms
0..*
0..*
Reservation2
-reservationID : int
-start : time_t
-end : time_t
-active : bool
-balance : double
+getStart() : time_t
+getEnd() : time_t
+isActive() : bool
+setActive(flag : bool)
+getBalance() : double
+setBalance(balance : double) : void
+Reservation()
+Reservation(time_t start, time_t end)
reservedBy
-roomID : string
-price : double
-bedCount : int
-desc : string
+getRoomID() : stri
+serialize() : string
+isAvailable(time_t
+getPrice() : double
+Room()
+Room(string room
+operation()
+reserve(Reservati
Account2
-guestAccounts : unordered_map<string, Account*>
+makePayment(string name, string addr, string email, CardType type, string cardNo, string exp, int cvv)
+addReservation(Reservation res)
+getInstance(string username) : Account2*
0..*
reservations
1
reservation

</td><td>
Visual Paradigm Professional(Josh Ibad (California State Univ., Fullerton))
createReservation
: Fascade
6: +makePayment(string name, string addr, string email, CardType type, string cardNo, string expirationDate, int cvv)
: ReservationHandler
1: reserveRoom(string roomID, time_t start, time_t end): double
: Hotel
4: +makePayment(string name, string addr, string email, CardType type, string cardNo, string expirationDate, int cvv)
1.1: hotelRoom = getHotelRoom(roomID)
: Account
hotelRoom : Room
3: create(name, addr, email, type, cardNo, expirationDate, cvv)
billingMethod : BillingMethod
1.1.1: create(time_t start, time_t end)
5: pendingRes = reservations.pop_back()
newReservation : Reservation
pendingRes : Reservation
1.1.1.1: reservations.push_back(newReservation)
1.1.2: push_back(newReservation)
2: create(billingMethod, pendingRes)
: Account
reservedBy : vector<Reservation>
newTransaction : Transaction

</td></tr>
</table>

Controller	Static View	Dynamic View
Option 2: Session (Use Case) Controller (Selected)	 <<Interface>> HotelHandler +reserveRoom(string roomID, time_t start, time_t end) : Reservation +getHotelRoom(string roomID) : Room* +addHotelRoom(double price, RoomType rType, BedType bType, int bNum, string desc) : string +getHotelRooms() : vector<string> +getAvailableHotelRooms(time_t start, time_t end) : vector<string> implements Hotel +reserveRoom(string roomID, time_t start, time_t end) : Reservation +getHotelRoom(string roomID) : Room* +addHotelRoom(double price, RoomType rType, BedType bType, int bNum, string desc) : string +getHotelRooms() : vector<string> +getAvailableHotelRooms(time_t start, time_t end) : vector<string> bedT hotelRooms 0..* 0..* Transaction mentDate : time_t ansaction(BillingMethod bill, Reservation res) reservation 1 1 billing Reservation -reservationID : int -start : time_t -end : time_t -active : bool -balance : double +getStart() : time_t +getEnd() : time_t +isActive() : bool +setActive(flag : bool) +getBalance() : double +setBalance(balance : double) : void +Reservation() +Reservation(time_t start, time_t end) reservedBy Ro -roomID : string -price : double -bedCount : int -desc : string +getRoomID() : string +serialize() : string +isAvailable(time_t start, time_t end) : bool +getPrice() : double +Room() +Room(string roomID, double price, RoomType roo +operation() +reserve(Reservation) 0..* reservations BillingMethod dNumber : string iration : string : int dHolder : string ngAddress : string ngEmail : string lingMethod(string name, string addr, string email, CardType ty, string cardNo, string exp, int cvv) cardType 1 <<Enum>> CardType -Visa -MasterCard -American_Express -Discover Account -guestAccounts : unordered_map<string, Account*> +makePayment(string name, string addr, string email, CardType type, string cardNo, string exp, int cvv) +addReservation(Reservation res) +getInstance(string username) : Account* implements <<Interface>> ReservationHandler +makePayment(string name, string addr, string email, CardType type, string cardNo, string exp, int cvv)	 +makePayment(string name, string addr, string email, CardType type, string cardNo, string expirationDate, int cvv) : ReservationHandler 1: +makePayment(string name, string addr, string email, CardType type, string cardNo, string expirationDate, int cvv) : Account 1.1: create(name, addr, email, type, cardNo, expirationDate, cvv) billingMethod : BillingMethod 1.2: pendingRes = reservations.pop_back() pendingRes : Reservation 1.3: create(billingMethod, pendingRes) newTransaction : Transaction addHotelRoom(double price, RoomType roomType, BedType bedType, int bedCount, string desc) : HotelHandler 1: addHotelRoom(double price, RoomType roomType, BedType bedType, int bedCount, string desc) : Hotel 1.1: create(price, roomType, bedType, bedCount, desc) newRoom : Room 1.2: push_back(newRoom) hotelRooms : vector<Room>
Design Model Reference	In document 09 - Design Model, at section '1 Static View', under subsection '1.1 Hotel Reservation system - Room and Reservation Management'. (Page 2).	In document 09 - Design Model, at section '2 Dynamic View', under subsection '2.2 addHotelRooms Sequence of Execution' (Page 4), and subsection '2.6 makePayment Sequence of Exeuction' (Page 8).

2.1.3 Selection and Rationale

Option 1 has been discarded because, although it centralizes all interaction into a single system-wide Facade controller, it also means that the singular controller will be highly coupled with all other Domain-layer components and classes. In the long-run this means higher coupling and it also means lack of scalability as more functions and classes are added in the Domain-layer components.

Option 2 has been selected because it elegantly permits the subdivision of the Domain-layer into a subset of low-coupling components or sub-packages that scales well when more functionalities and classes are added. By making each system level message a single interface method relayed to a single component, rather than use case messages that then send system-level messages scattered all over the domain layer, we have have a better design with the Session controllers (use-case level controllers with handlers for each component).