1 Integration Strategy

1.1 Entry Criteria

Before starting the integration phase, we have to meet certain constraints on different aspects of the project.

All components must have been unit tested The integration procedure should start after these percentages of completion are reached:

- **100%** of DatabaseController functionalities
- At least **90%** of Customer subsystem
- At least **80%** of Car subsystem
- At least **50%** of Employee subsystem

1.2 Elements to be Integrated

As it can be seen from the Design Document, the whole system can be divided into three major subsystem:

- **Customer subsystem**, offering functionalities to exploit the services provided by the car sharing company
- **Employee subsystem**, offering functionalities for company's employee to manage the system
- **Car subsystem**, offering functionalities to dialog with cars

Each of this subsystems is divided into submodules encapsulating the components related to a group of functionalities in this way:

- **Customer subsystem**:
  - *ReservationController*: offers the user functionalities to manage a reservation
  - *BalanceController*: offers the user functionalities to manage his balance
  - *AccountController*: offers the user functionalities to manage his account
  - *RegistrationController*: offers the user functionalities to register to the system
  - *ReportController*: offers the user functionalities to notify issues
- **Employee subsystem**:
  - *SafeAreaController*: offers the employee functionalities to manage safe areas

- *AssistanceController*: offers the employee functionalities to manage needy cars

- **Car subsystem**:
  - *CarController*: offers the system functionalities to send commands to the car
  - *RideController*: offers the CarApp the functionalities to manage a ride
  - *BillController*: offers functionalities to calculate bills
  - *FaultController*: offers the CarApp the functionalities to manage faults
  - *FleetController*: offer the CarApp the functionalities to join and detach from the fleet

The elements to be integrated are the components of these submodules

1.3 Integration Testing Strategy

We will follow a functional grouping approach, grouping components in the submodules presented above. We will first integrate functional groups of Customer subsystem, as these are the most critical, then we will proceed integrating functional groups of Employee subsystem and Car subsystem.

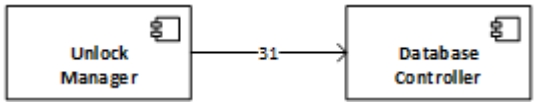
For each functional group we will follow a bottom-up approach, and once integrated all submodules of a subsystem, we will integrate the subsystem with its dispatcher and web application. This will allow us to focus initially on binary integrations and to start testing complete functionalities as we proceed.

1.4 Sequence of Components/Function Integration

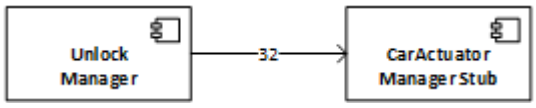
1.4.1 Customer subsystem

Reservation Controller

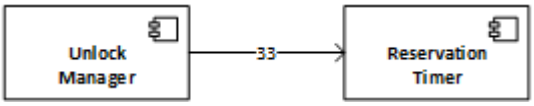
First we integrate the UnlockManger with the DatabaseController



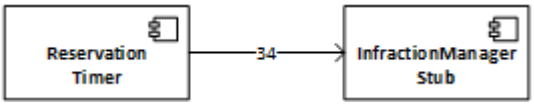
Then we integrate the UnlockManager with the CarActuatorDriver



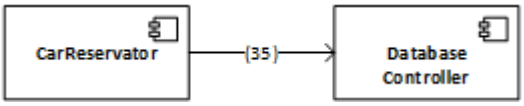
At last we integrate the UnlockManager with the ReservationTimer



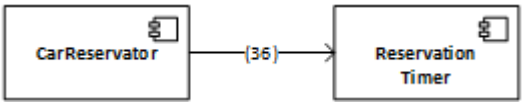
We proceed integrating the ReservationTimer with the InfractionManager-Driver



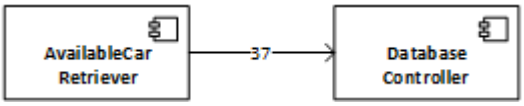
Now we integrate the CarReservator with the DatabaseController



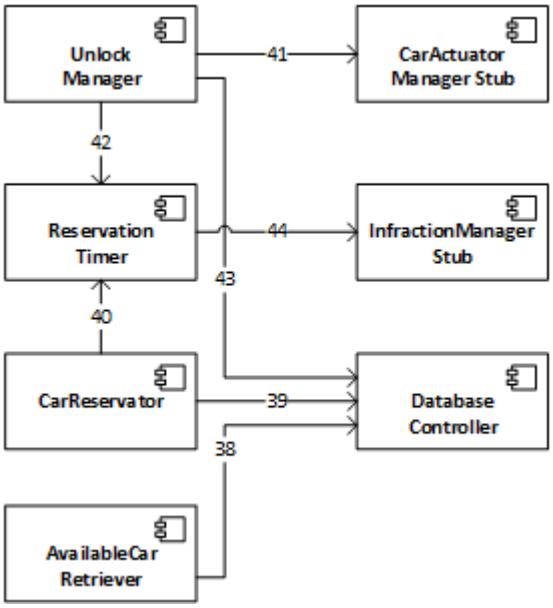
We integrate it also with the Reservation Timer



To complete the subsystem integration, we integrate AvailableCarRetriever with DatabaseController

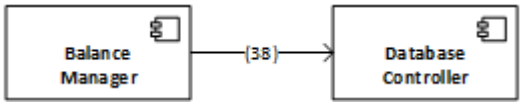


This is the final integrated ReservationController module

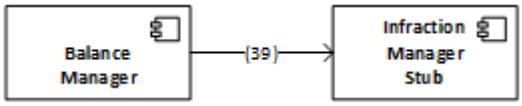


BalanceController

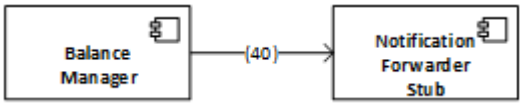
To integrate the components of this module we start from integrating BalanceManager with DatabaseController



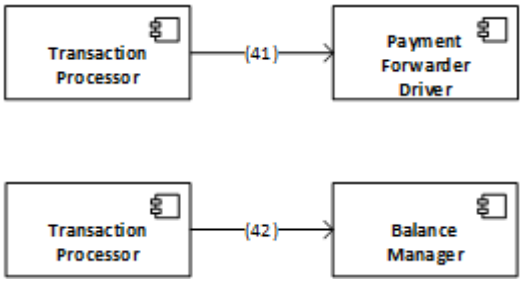
We integrate it also with the InfractionManager



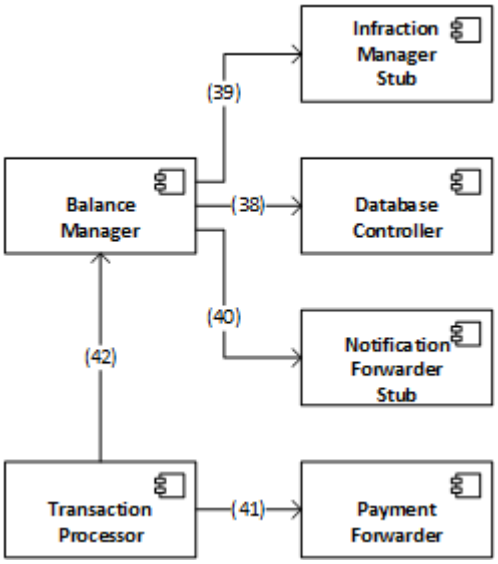
And with the NotificationForwarder



Then we integrate TransactionProcessor with both BalanceManager and PaymentForwarder

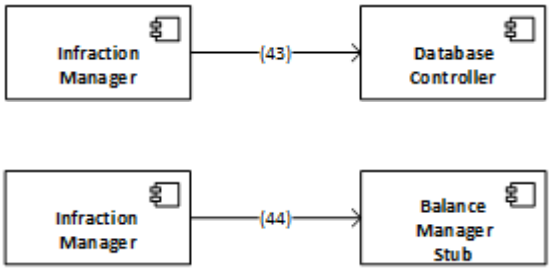


This is the final integrated BalanceController module

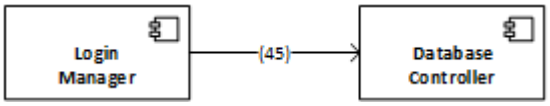


AccountController

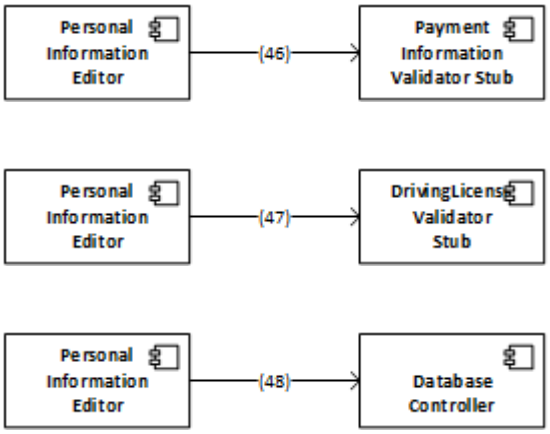
We start integrating InfractionManager with DatabaseController and BalanceManagerStub



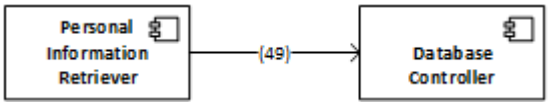
The second step is integrating LoginManager with DatabaseController



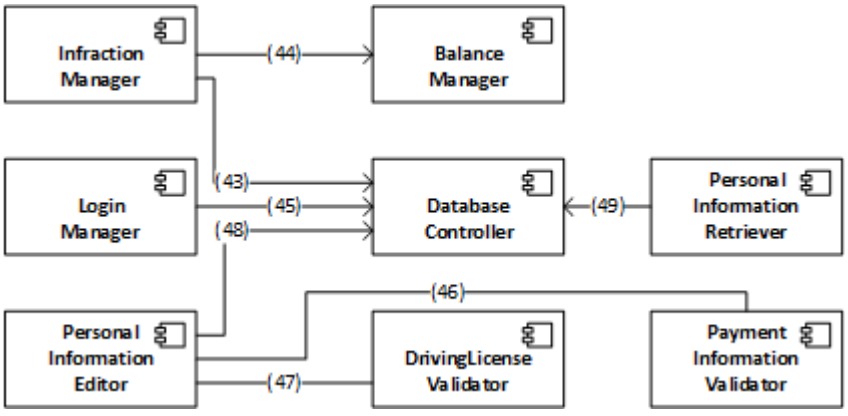
After that we integrate PersonalInformationEditor with PaymentInformationValidator, DrivingLicenseValidator and DatabaseController



The last integration is the one between PersonalInformationRetriever and DatabaseController

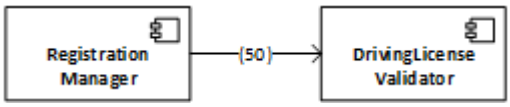


This is the final integrated AccountController module

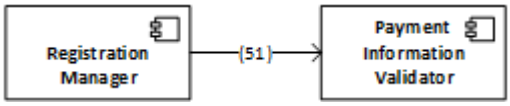


RegistrationController

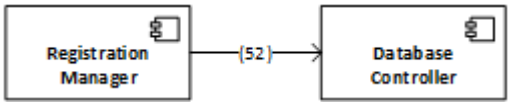
We start the integration procedure from the integration between the RegistrationManager and the DrivingLicenseValidator



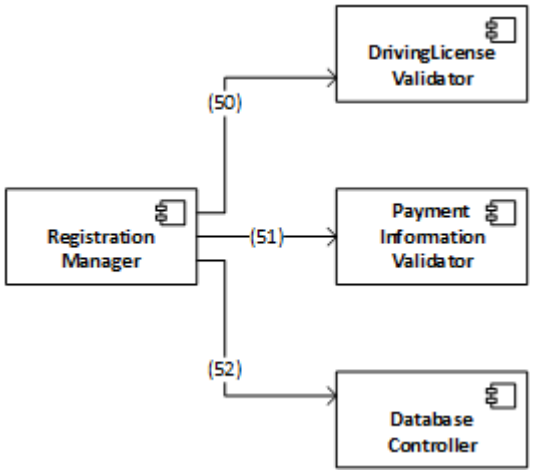
As second step, we integrate the RegistrationManager with the PaymentInformationValidator



In the end we integrate RegistrationManager with DatabaseController

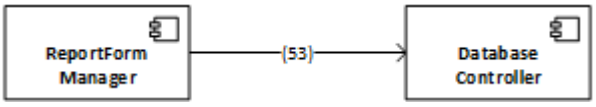


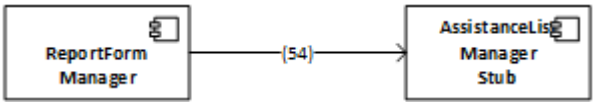
This is the final integrated RegistrationController module



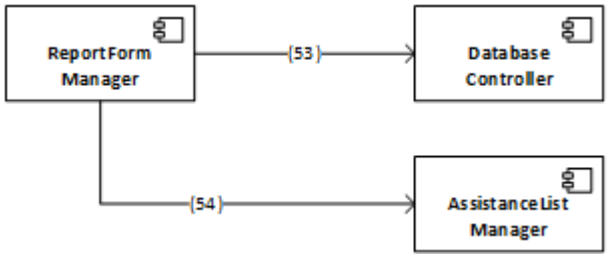
ReportController

For this module we integrate the ReportFormManager and both the DatabaseController and the AssistanceListManagerStub



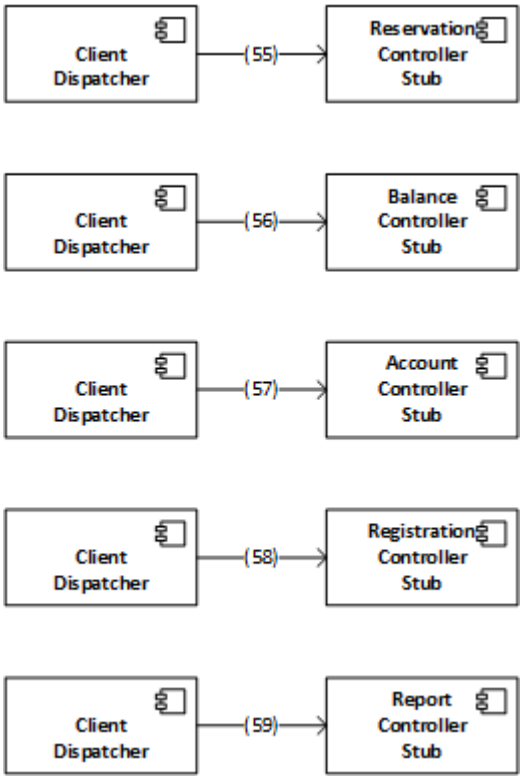


This is the final integrated ReportController module

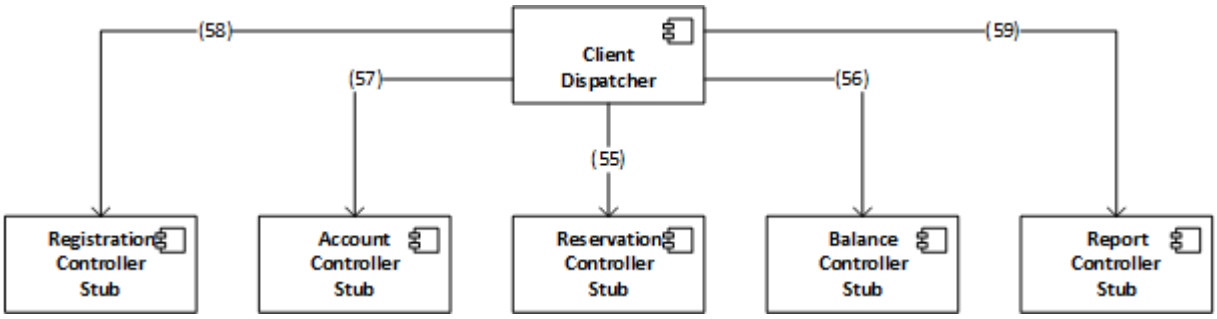


ClientDispatcher

We will integrate the ClientDispatcher with each one of the previously integrated modules



This is the final integrated module


(58)
Client
Dispatcher
(59)
(57)
(56)
(55)
Registration
Controller
Stub
Account
Controller
Stub
Reservation
Controller
Stub
Balance
Controller
Stub
Report
Controller
Stub