Distributed Systems

Breyne Emile, Engels Willem

Design overview

The client can only communicate with two remote components: the SessionManager and a Session. It does so trough the interfaces ISessionManager and ISession. The SessionManager is meant to create a new Session for a specific client. The SessionManager also provides the life cycle management of the Sessions. it can open, store and remove a Session. A Session can either be a ReservationSession or a ManagerSession. These Sessions provided the required functionality to the client, so the client can communicate with the system.

Another remote component is the NamingService. A Session communicates with the NamingService through the interface INamingService, and this component provides the registration of all the different car rental companies into the system.

A Session will call upon the NamingService to get a specific CarRentalCompany. A ManagerSession specifically will call upon the NamingService to register and unregister different CarRentalCompanies.

Remote and Serializable classes

SessionManager and Session are remote classes. These two components are remote because they interact with the client and should thus be deployed on different nodes. Because these components shouldn't be deployed on the node of the client.

The component NamingService is also Remote. It is chosen as a remote component in stead of a component that runs on the same host as the Sessions and the SessionManager. In a small system, with just a few car rental companies, it should be no problem to deploy this all on the same node. But with a large rental agency with a lot of different rental companies, it might be better to let the NamingService run on its own node. Having it as a remote component have a negative impact on the performance, but it is useful for future extensions to the system.

The component CarRentalCompany is also a remote component. This component should be remote because there are many car rental companies within a single car rental agency. These are all in different locations and should therefor be deployed on different nodes and be accessed remotely.

There are three components that are serializable: CarType, Reservation and Quote. These are the only components, other then the components that are already remote, that get transferred trough the system as data. So no other components should be serializable.

Hosts of remote objects

The components SessionManager and Session are located on the same hosts, because the SessionManager will initialise a Session. This can't be done by a component that is on a different host.

The CarRentalCompany is on a different host then the SessionManager and the Session components. The CarRentalCompoany represents a single company that is part of a bigger agency. The host that contains the Sessions could be seen as the central host of the agency, and all individual rental companies are represented by different hosts that each run their own CarRentalCompany.

Remote objects registered via RMI registry

The Sessions, SessionManager and NamingService are the three remote object that are registered via the RMI registry. because the run on nodes that are unique, and therefor they are easily registered into the RMI registry.

CarRentalCompany is not registered via the RMI registry, because each CarRentalCompany runs on a different node. Binding them to an RMI registry might cause some concurrency problems. All active CarRentalCompanies are stored in the NamingService.

Life cycle management of sessions

The life cycles of the different sessions is managed by the component SessionManager. A client has remote access to the SessionManager and can ask it to create a ReservationSession or a ManagerSession. The SessionManager will then first check if the client already has an active session registered to his name. If this is the case, this existing session is returned. Otherwise a new Session is created and registered via the RMI registry.

The SessionManager can also remove these Sessions from the RMI registry.

Synchronisation

Synchronisation is necessary at the method confirmQuotes in the component ReservationSession and for the methods registerCarRentalCompany and unregisterCarRentalCompany in the component ManagerSession. It is possible these places can become a performance bottleneck, but they have to be synchronised. There is a great risk of concurrency problems at these places.

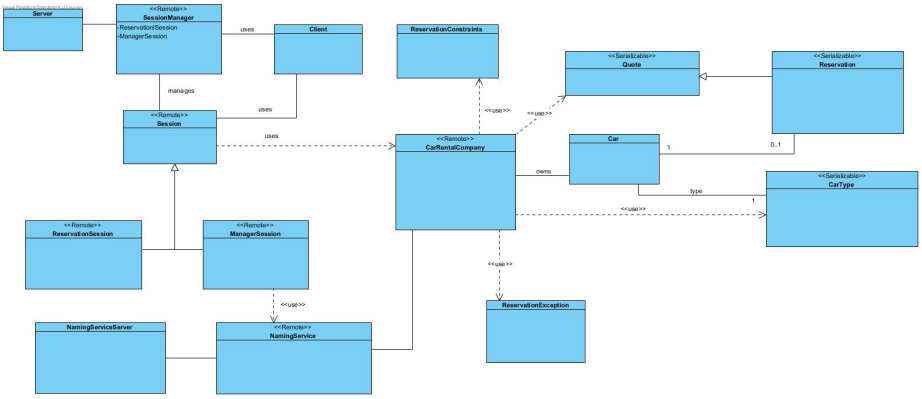

Figure 1: Class diagram

Diagram1.jpg

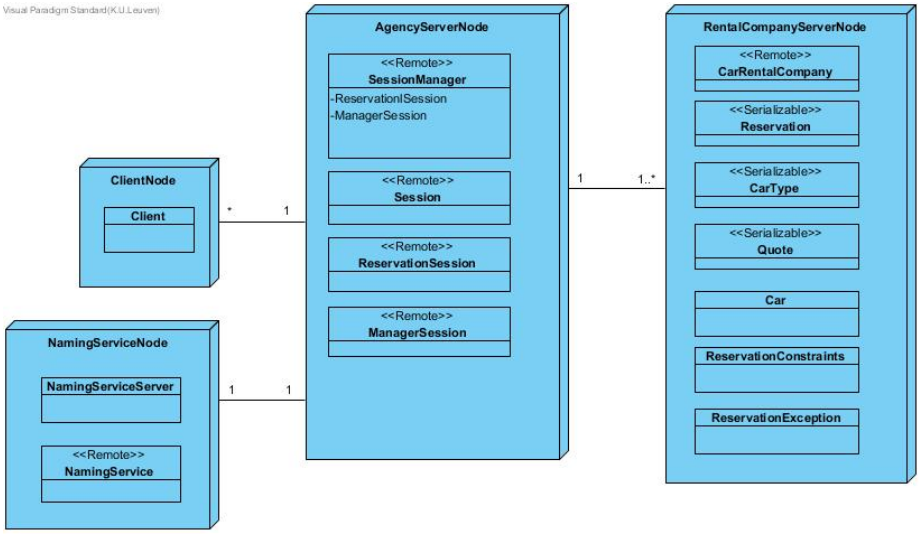

Figure 2: Deployment diagram

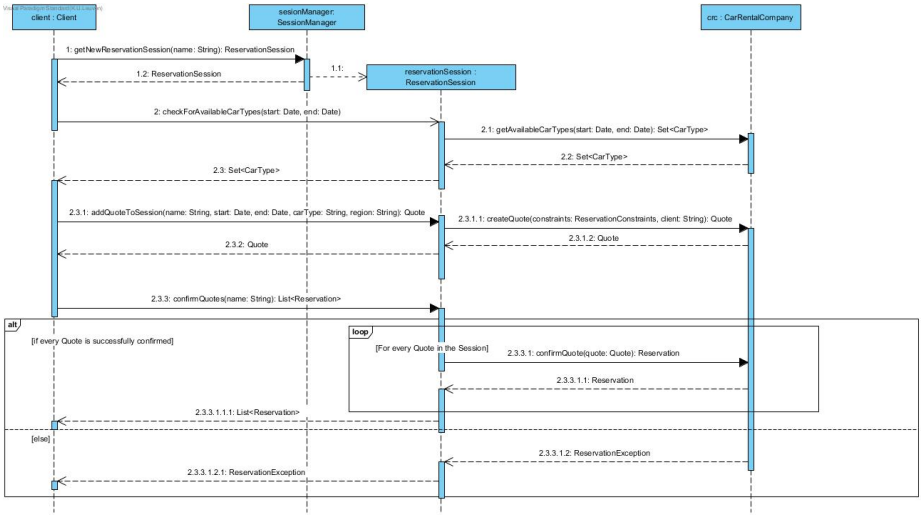

Figure 3: Sequence diagram