INN374 Assignment 2

Service-Oriented Architecture

for Automotive Maintenance Activities

Student Name: Yinlong Sun

Student Number: n8615136

Brief description of system:

The system includes one windows application, 3 windows services, 2 linux services and 2 database(one for SQLServer and one for MySQL).

Windows application is created by vs windows form application, written by C#. The application support a GUI to client, which simplify the operation. Application also can connect other 5 services. This application is designed to sales role.

Warehouse service can do a lot sql operation in terms of insert, update, delete and get items from database.

Customer service can act as a customer role, it can generate a random customer name, request and also can reply the decision to sales.

Supplier service can return a random number to present the delay time.

Insurance service will return a random type of insurance.

Garage service will store the appointment information in database.

URL(s)

http://fastapps04.qut.edu.au/n8615136/customer.html

http://fastapps04.qut.edu.au/n8615136/warehouse.html

http://fastapps04.qut.edu.au/n8615136/insurance.html

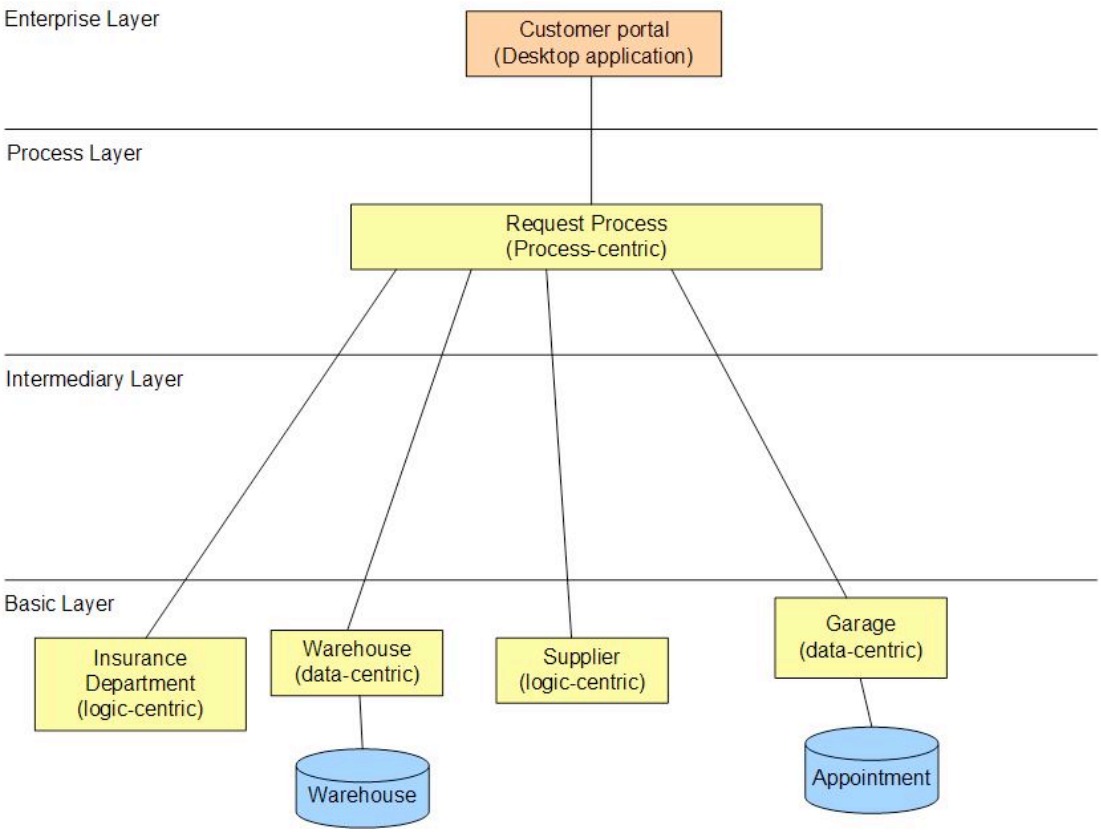
http://fastapps04.qut.edu.au/n8615136/supplier.html

http://fastapps04.qut.edu.au/n8615136/garage.html

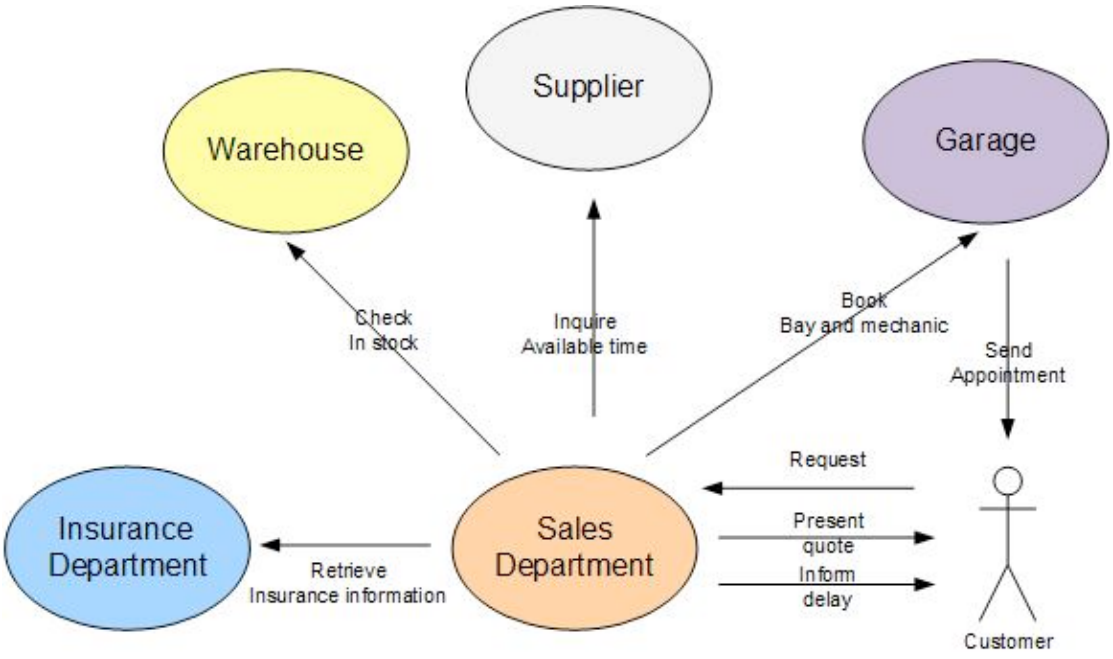
Application's file name: Sales.exe

Due to it is C# windows form application, it need to work under framework. All of necessary files are hold in one folder named Sales Application. Open the folder and click Sales.exe file, then just click the button can make it run.

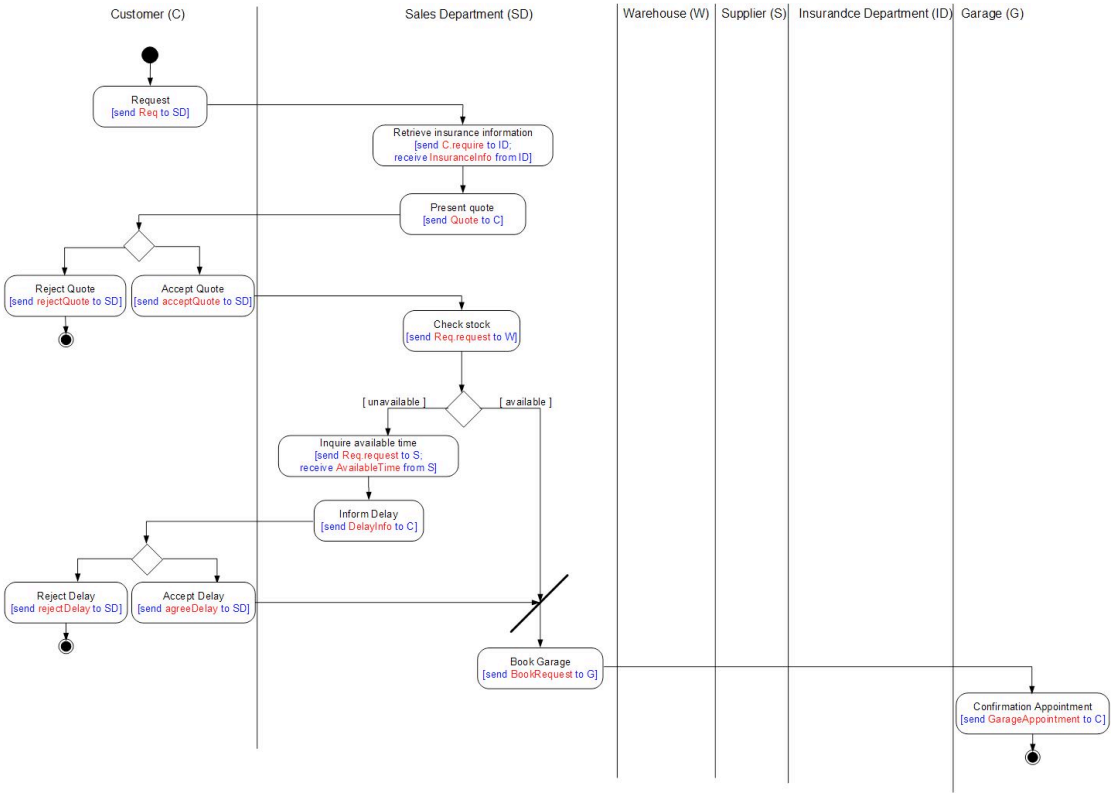
Logical Layered Architecture



Interaction Diagram



Choreography Model



SOA analysis

I chose to follow bottom-up service analysis, because in this case, each entity can be assumed independently. It is effectively to develop an independently item without considering external relative. Therefore, it is quickly to complete each entity and then through an interchange to connect them. The advantage of this methodology is that developer do not need to consider the entire system, which can save time on designing framework.

For top-down service analysis, it need design a big picture for the product and assign the most reasonable function to each individual. This methodology is good for operation of system, it makes all parts work in whole system, and with high work efficiency.

However, in assignment case, each entity is required and no part is redundant. Therefore, I think bottom-up is more suitable for this system, which is more effective.

Assumptions

I simplify the customer role, make the service just return random reply to sales. Also I did the same to supplier service, it just return random delay time to sales, the purpose is to make the whole simulation system running well. In customer request, each time can only require one item. For insurance service, it also just return a random string for insurance information.

Aspects of service orientation

In warehouse service, it is not only a data-centric, but logic-centric, which can check the item whether available or not in stock. Warehouse is a place to store goods, so it has the same function as the database. It can get stock or sale stock, which like the insert, update and delete method in database. Also the service can be reusable.

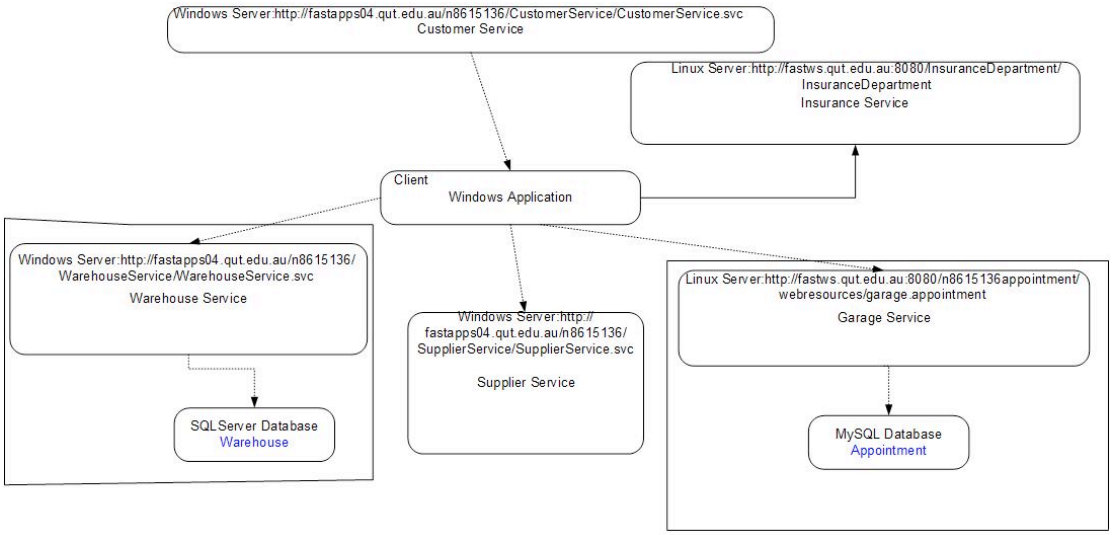
Supplier service, logic-centric service.

Customer, logic-centric service.

Insurance, logic-centric service.

Garage, data-centric service.

Deployment Diagram



Service Table

Name	Platform	Language	Technology	Complexity	Author
Sales	Windows	C#	Windows Form	240 lines	Yinlong
Customer	Windows	C#	WSDL	150 lines	Yinlong
Warehouse	Windows	C#	WSDL	200 lines	Yinlong
Supplier	Windows	C#	WSDL	20 lines	Yinlong
Insurance	Linux	JAVA	WSDL	20 lines	Yinlong
Garage	Linux	JAVA	RESTful		Yinlong

*The Insurance service has some problem may return null value, I've fixed it but the glassfish is down and I can not deploy it, which may cause some confusion when using application and insurance return a null value. However, it will not affect the whole operation running, if get null value, just ignore it.