Faculty of Computing, Engineering and the Built Environment

BIRMINGHAM CITY University

Undergraduate Programme
Academic Year 2014-2015
Coursework: Team Project

Module:	*CMP2515 Software Design UG2*
School:	**Computing, Telecommunication and Networks**
Module Co-ordinator:	**Professor Zhiming Liu**
Setup Date:	**20/02/2015**
Submission Date:	**21/04/2015**
Team Number:	T15
Team Leader:	Lucy Shone
Team Members:	Bradley Smith, Aleksey Ziarniecki, Imaad Yasin, Jamie Uddin, Mohammed Yaseen, Gurpreet Singh

Instruction to Students:

The final report to submit should contain this page as the cover page, with the above details of the team filled in.

As part of project management, each team should have a weekly meeting. A weekly project diary should be maintained to record the attendance of the meetings by team members, together with brief notes about the weekly project tasks allocations and how well individual team members meet the deadlines of their project tasks.

The final project report should include the weekly dairy in the part of the project management.

Contents

I. Initial requirements understanding

1) Discuss the nature of the Bvis Car Hire Company. Why is an object-oriented development applicable to this company? Refer to chapter 3 of the course notes. Elaborate the problem description as necessary to support your analysis.

The nature of the company is to supply a car hire service to its customers based on a daily hire rate. It currently used a paper based manual system, however this type of record system puts the company at a higher risk of lost paperwork and records. A computer system would help store all of this information digitally, therefore the company would be at a lower risk of losing records and paperwork, and also potentially being able to retrieve lost data.

For this system an object-oriented development will be used. Object oriented is a style of programming that uses objects. Objects are used to store data, such as number data (integers) and word data (strings). This would be applicable to this company because of the type of data being stored. Figure 1.1 shows the process outline of user input when using object-oriented development:

批注 [ZL1]: Not only a style of programming

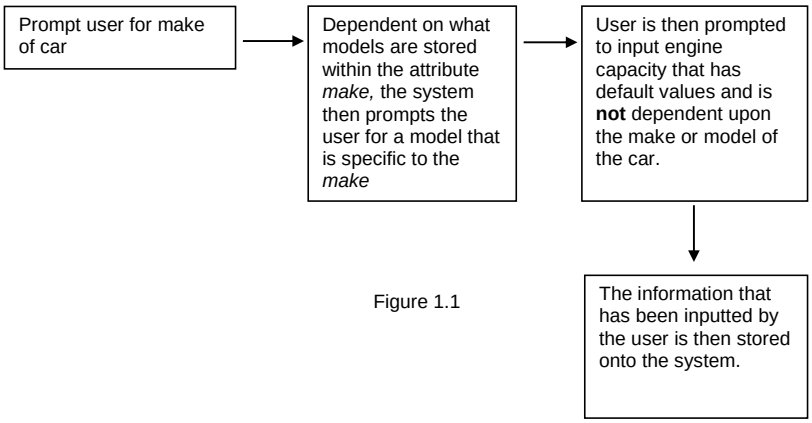


Figure 1.1

Five attributes of a complex system:

1. 'Complexity takes the form of a hierarchy, whereby a complex system is composed of inter- related subsystems that have in turn their own subsystems, an so on, until some lowest level of elementary components is reached.'

This enables a company to be split into multiple departments. The departments will then contain 'sub-departments' and they can view data and information that is specific to that department, as opposed to having an overload of data and information. For example: the company could contain a department or section of the system that is called 'employees', this section will then include different sub-sections, such as: 'management' or 'administration' for instance.

2. 'The choice of what components in a system are primitive is relatively arbitrary and is largely up to the discretion of the observer of the system.'

This enables a company to ensure different departments only have access to stuff they need. The person in charge of the system is the one who decides who has access to what, so an

example of this would be the sales assistant that deals with customers does not need to have access to information regarding the payment of all staff. So this attribute enables the limitation of information to those who do not need it.

3. ‘Intracomponent linkages are generally stronger than intercomponent linkages. This fact has the effect to separating the high-frequency dynamics of the components - involving the internal structure of the components - from the low-frequency dynamics - involving interaction among components.’

All information in a system is used by other departments meaning that no duplicate data needs to be added, an example of this is all departments that deal with the sale process and end product need to know the customerID. This means that this information must be shared across all sections and is no unique to a certain department.

4. ‘Hierarchic systems are usually composed of only a few different kinds of subsystems in various combinations and arrangements.’

An example of this would be when a customer is returning a hired car, instead of having to enter all of the details again only the new details would be needed like the new mileage of the car. This then results in much less duplicate data.

5. ‘A complex system that works is invariably found to have evolved from a simple system that worked. A complex system designed from scratch never works and cannot be patched up to make it work. You have to start over, beginning with a working simple system.’

Every complex system must start from somewhere, each system must have a starting backbone, which can be changed and eventually made into a complex system. An example of this is when new programs are made the developers have a clear plan of what the system must include and start with all the basics to make the program work and easily upgradable showing the fact that a complex system evolves from a simple system that works.

(*Please continue to next page…)*
2) System functions according to the guidelines in section 4.1 of the course notes.

Basic Functions:

Table 1.1 shows the basic functions of the system. These have been grouped as ‘basic’ because they are the functions the system should be able to do at a minimum based on the problem description.

Ref#	Function	Category
1.1	Register a new customer onto the system	Hidden
1.2	Record that a particular car has been hired	Evident
1.3	Record that a particular car has been returned	Evident
1.4	Calculate the cost based on the daily hire rate	Evident
1.5	Display the appropriate details, and print out a receipt	Evident
1.6	Log a completed hire	Hidden
1.7	Record a service for a particular car, together with the date of the service, the type of service and the name of the mechanic responsible for the service	Evident
1.8	Add a new car to the fleet	Evident
1.9	Add a mechanic who has joined the company	Hidden

Table 1.1

Data removal functions:

Table 1.2 shows other functions that have been placed in groups other than ‘basic’. These functions have been grouped as ‘data removal’. All of the following functions remove data that will be stored in the computer system, however none of them affect the ‘basic’ functions, therefore they are *frill functions.*

2.1	Remove a customer	Hidden
2.2	Delete a car that is no longer in the hire fleet	Hidden

2.3	Remove the details of a mechanic who has left the company	Hidden

Table 1.2

Data extraction functions:

Table 1.3 shows functions that aren't categorised as 'basic' or 'data removal'. The remaining functions have been grouped as 'data extraction' and rely on the basic functions to display data when required by the actor. They don't however affect the basic functions, and therefore, are *frill functions.*

3.1	Determine if a particular car is due for a particular service	Evident
3.2	List the information (history) about all hires for a specified car	Evident
3.3	List the information (history) about all services that a specified car has had	Evident

Table 1.3

(*Please continue to next page…)*

3) Identify the ESSENTIAL use cases.

Essential use cases highlighted in bold.

1. Register a new customer.
2. Record that a particular car has been hired.
3. Record that a particular car has been returned.
4. Calculate the cost based on the daily hire rate.
5. Display the appropriate details, and print out a receipt.
6. Log a completed hire.
7. Record a service for a particular car, together with the date of the service, the type of the service, and the name of the mechanic responsible.
8. Remove a customer.
9. Add a new car to the fleet.
10. Delete a car that is no longer in the hire fleet.
11. Add a mechanic who has joined the company.
12. Remove the details of a mechanic who has left the company.
13. Determine if a particular car is due for a particular service.
14. List the information (history) about all hires for a specified car.
15. List the information (history) about all services that a specified car has had.

After reading the problem description for Bvis Car Hire Company, specific use cases were chosen that would be essential to the company. Each individual use case plays a critical part within the business and they are used to describe the possible sequences of an event.

10 uses cases were selected out of a total of 15. The 10 that were chosen were described to be the main and important events that will occur in the car hire company.

(*Please continue on to next page…)*

4) Write an expanded version for each of these use cases.

Firstly identifying the use cases, from these, select a few use cases that we think are crucial to the use of the system based on the problem statement.

Register a new customer

Use case: Register a new customer

Actors: Customer (initiator) and the sale assistant

Purpose: To identify that a new customer has joined the company

Overview: The customer wants to register as a new customer. The sale assistance asks the customers for their details such as name, telephone number and address on a form. On completion after the customer has filled his/her details the customer leaves.

Typical course of events

ACTOR ACTION	SYSTEM RESPONSE
1. This use case begins when a customer wants to register with the company. 2. The actor provides the information required to register with the sales assistant: name, telephone number, and address.	
	3. Information is saved onto the system and the customer is now registered with the company.

Alternative courses of action:

- 2: Invalid data entry input. Prompts sales assistant to re enter information. Indicate error.
- 3: System failed to save information. Prompts sales assistant to re enter customer information.

Record that a particular car has been hired

批注 [ZL2]: Simplify

Use case: Record that a particular car has been hired

Actors: Customer (initiator) and the sale assistance

Purpose: To identify that a car has been hired

Overview: The customer wants to hire a car. He/she would have to provide information such as their name, address, telephone number and driving license before they can hire the car. The beginning and end of the hire are recorded as well. This is to identify what car has been hired. Details such as the registration number (unique), make, model, engine capacity, hire class (1 – 6) and the date of the registration.

Typical course of events

ACTOR ACTION	SYSTEM RESPONSE
1. This use case begins when a customer wants to hire a car. 2. Information of the customer is provided.	
	3. System checks to see if the customer is registered on the system.

3. Customer states make, model and engine capacity of the car that they want to hire and the date of hire.	
	5. System checks to see if the car is available for hire on that specific date.
6. Sales assistant completes hire enquiry.	
	6. System registers the hire of the car onto the system.

批注 [ZL3]: Could be more informative

Alternative courses of action:

- 3: Customer is not registered and has to be registered before hiring a car.
- 5: Date of hire or car is unavailable. Prompts the sales assistant to enquire a different date or model of car that is available.

Record that a particular car has been returned

批注 [ZL4]: Simplify the name

Use case: Record that a particular car has been returned

Actors: Customer (initiator) and the sale assistance

Purpose: To identify that the car has been returned after it has been hired

Overview: The customer wants return the car they have hired. When the customer comes to return the car the sales assistance checks to see if the same car is being returned.

Typical course of events

ACTOR ACTION	SYSTEM RESPONSE
1. Use case begins when a customer returns with the hire car. 2. Sales assistant records mileage and date of return for a particular car. 3. Sales assistant completes hire agreement file.	
	4. System updates the hire agreement file.

Alternative courses of action:

- 2: Incorrect data input. Indicate error.

Calculate the cost based on the daily hire rate

Use case: Calculate the cost based on the daily hire rate.

Actors: Sales assistant (Initiator)

Purpose: capture a specific hired car and its total cost based on a daily rate so that a receipt can be produced.

Overview: The car details with the time hired are pulled up and calculated so that the total cost of the hire has been worked out. A receipt is then produced.

Typical course of events

ACTOR ACTION	SYSTEM RESPONSE

1. Car is returned after hire period. 2. Sales assistant will then enter the return date.	
	3. System uses the return date to calculate a cost using: number of days of hire * daily hire rate. 4. System displays a final cost that the customer will need to pay.

Alternative courses of action:
- 2: Invalid date is inputted. Indicate error.
- 3: Invalid cost produced. Indicate error.

Display the appropriate details, and print out a receipt

Use case: Display the appropriate details, and print out a receipt.

Actors: Sales Assistant (initiator), Costumer

Purpose: Display the cost and print out a receipt

Overview: A customer's cost for hiring a car based on the daily hire rate is displayed.
The cashier prints out a receipt for the costumer. Once completed the costumer leaves with a receipt.

Typical course of events

ACTORS ACTION	SYSTEM RESPONSE
1. The use case begins with the sales assistant retrieving specific details of a hire off the system.	
	2. The system retrieves the details that are stored and displays them to the sales assistant.
3. If the customer agrees to the price, then the sales assistant will print out the receipt with the hire details and cost on.	
	4. The system then prints out a receipt.
5. Customer leaves with the receipt.	

Log a completed hire

Use case: Log a completed hire.

Actors: Sales assistant (Initiator)

Purpose: A record of each completed hire with the correct information is kept.

Overview: A sales assistant keeps a record of the hire dates, customer's details and the payment due.

Typical course of events

ACTOR ACTIONS	SYSTEM RESPONSE
1. The sales assistant will log into the system in order to log a complete hire.	

	2. The details of that specific hire will be displayed.
3. The sales assistant can amend these details and add a return date and mileage to the file. 4. The sales assistant then saves the file.	
	5. The system the updates the file.

Alternative courses of action:

- 1: Incorrect details input. Indicate error and prompt to re-enter user details.
- 2: No hire agreement file is located. Indicate error.

Determine if a particular car is due for a particular service

Use case: Determine if a particular car is due for a particular service.
Actors: Sales assistant (Initiator)
Purpose: Checking to see if a car needs servicing by a mechanic
Overview: A car is serviced depending on the miles it's done and a record is kept

Typical course of events

ACTOR ACTION	SYSTEM RESPONSE
1. Sales assistant logs onto the system. 2. Uses the system to retrieve details and all previous hires for a specific car.	
	3. System reacts actor by searching its system for the required car details.
4. Sales assistant checks current mileage for the car. 5. If the mileage is at 6,000 the sales assistant will book it in for a minor service. 6. If the mileage is at 12,000 the sales assistant will book it in for a major service. 7. If it is under 6,000 or already been serviced, then it doesn't require service and the sales will log off the system.	

Record a service

Use Case: Record a service

Actors: Sales Assistant (initiator)

Purpose: To record a car service onto the system.

Overview: A car returns from a service. The sales assistant records the type of service, the details of the service and the mechanic who performed the service onto the computer system.

Typical course of events

ACTOR ACTION	SYSTEM RESPONSE
1. Use case begins when a car returns from a minor or major service. 2. Sales assistant records the make, model and registration number.	

	3. System carries out a check to ensure the car is registered onto the system already.
4. Sales assistant records the type of service that has been performed on the car and the mileage at the time of service. 5. Sales assistant records the date of the service and the mechanic who performed the service. 6. Sales assistant indicates to the system that the record is completed.	
	7. System logs the completed record.

Add a new car to the fleet

Use case: Add a new car to the fleet.

Actors: Sales assistant (initiator)

Purpose: Add a new car to company's system for customers to hire.

Overview: The sales assistant will add the details of the new car on the system such as registration number, make, model, engine capacity and the hire class (1 - 6). A new car will be added to the company's system.

Typical course of events

ACTOR ACTIONS	SYSTEM RESPONSE
1. The use case beings with a sales assistant prompting the system to add a new car	
2. The sales assistant will input the details of the car such as registration number, make, model, engine capacity and the hire class (1 - 6)	
	3. The system will verify that all of the details are correct.
	4. The system will save the details of the car.

Delete a car that is no longer in the hire fleet

Use case: Delete a car that is no longer in the hire fleet.

Actors: Sales assistant (initiator)

Purpose: Delete a particular car from the company's system.

Overview: The sales assistant will input the details of the car to be deleted. Once the system displays the car the sales assistant will delete the car. The system will save that the car is no longer on the system.

Typical course of events

ACTOR ACTIONS	SYSTEM RESPONSE
1. The use case beings with a sales assistant wanting to delete a car.	

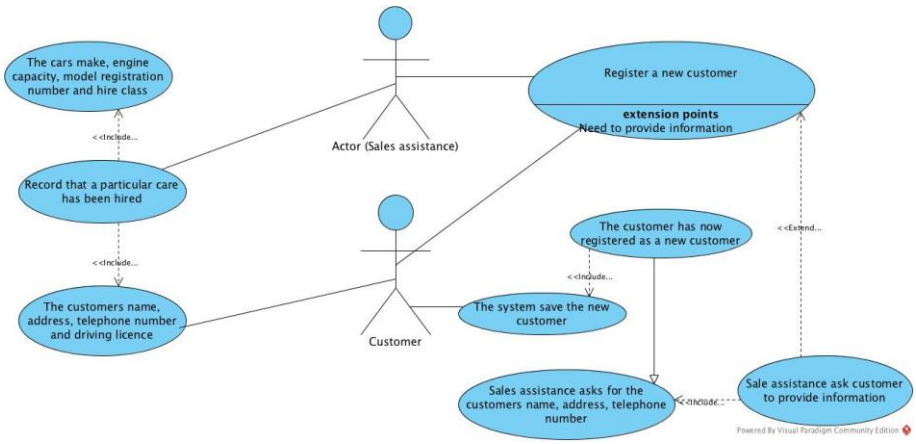
2. The sales assistant will enter the details of the car such as registration number, make, model, engine capacity and the hire class (1 - 6).	
	3. The system will display a car that matches the details of the data.
4. The sales assistant will then prompt the system to delete the car.	
	5.The system will save with the car being deleted.

To write the expanded version for each of the use cases we had to begin with writing high level use cases for each of the essential use cases that we choose. A high level use case would give us an overall understanding of the process. Once we had done the high level use cases we moved on to doing the expanded use cases which required us to analyse the high level use case and then write the expanded use cases.

批注 [ZL5]: Very good use case decriptions

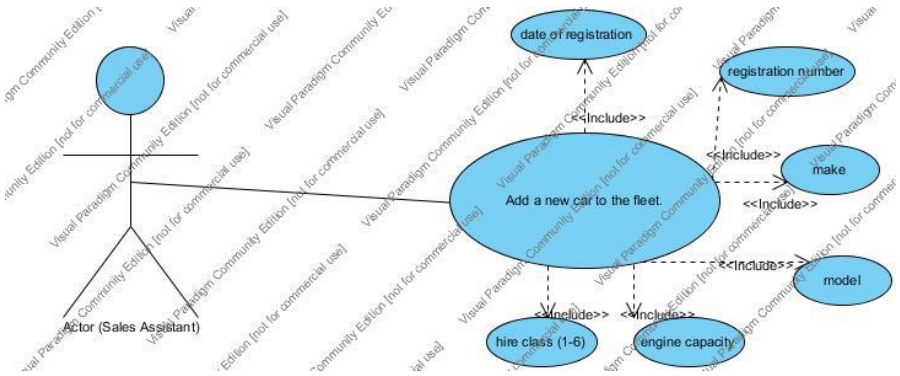
(*Please continue to next page…)*

5) Use case diagrams that show the relationships between the actors and the use cases.

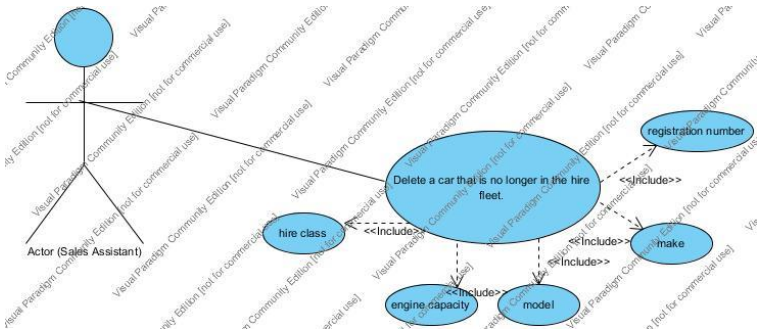


Use case diagram 1 – Record a hire and register and new customer.

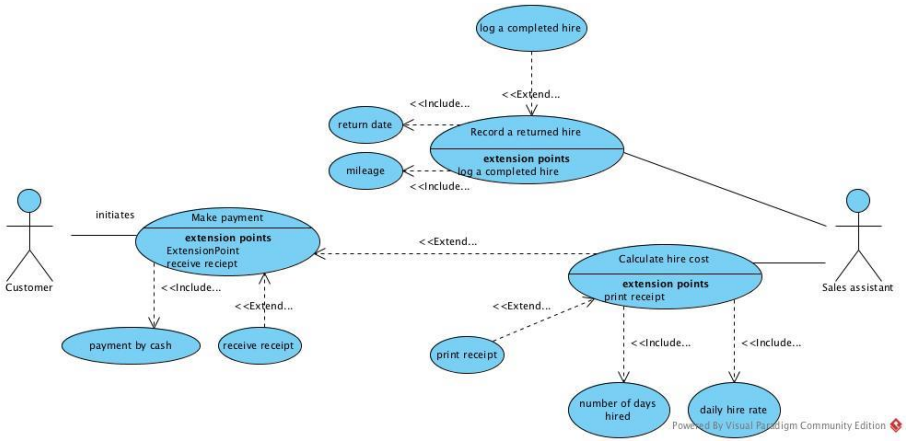
批注 [ZL6]: Many of the use cases in the diagram are not explained or described



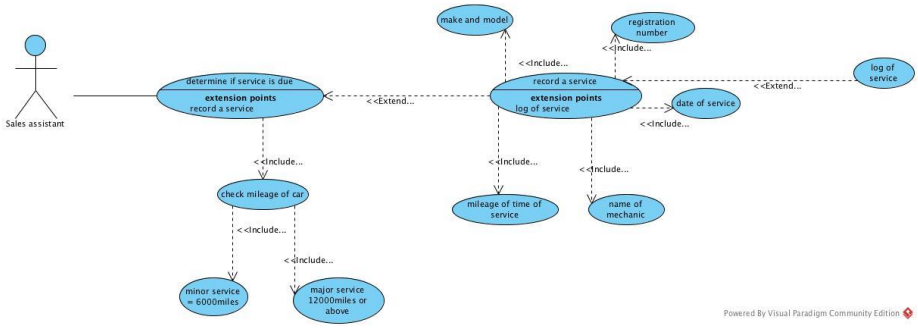
Use case diagram 2 – Add a new car to the fleet.



Use case diagram 3 – Deleting a car from the fleet



Use case diagram 3 – Recording a returned hire and calculating a cost.



Use case diagram 4 – Determine if a car is due for a service, and record a service.

Discussion for choice of use cases

Recording a returned hire and calculating a cost
The relationship in this use case diagram involve the use cases ‘recording a returned hire’ and ‘calculating a cost’. These 2 uses cases have a relationship because calculating a cost is due right after a car has been returned. The relationship between the uses cases and actors can easily be shown on this diagram.

In this diagram the sales assistant is associated with 2 use cases, one of which is called ‘record a returned hire’, this use case has an extension point of log a completed hire. ‘Return date’ and ‘mileage’ were included from this use case, which was extended from log a completed hire.

The second use case was ‘calculate hire cost ’, this extended from ‘’print receipt and had an extension, which was also print receipt and includes ‘number of days hired’ and ‘daily hire rate’ Another extended use case from this point is ‘make payment’. This had an extension point of ‘receive receipt’, which included ‘payment by cash’ and an extended from ‘receive receipt’. The ‘make a payment’ use case is associated with the customer.

Add a new car to the fleet
For the use case add a new car to the fleet a use case diagram was created in visual paradigm to show the relationships between the actor and the use case. An actor tool was used along with the association tool to show the association between the use case add a new car to the fleet and the actor. The use case diagram shows all the details that need to be included for add a new car to the fleet that are date of registration, registration number, model, make, engine capacity and hire class (1 - 6) that were added by using the include use case tool.

Delete a car that is no longer in the hire fleet
For the use case delete a car that is no longer in the hire fleet a use case diagram was created in visual paradigm to show the relationships between the actor and the use case. An actor tool was used along with the association tool to show the association between the use case delete a car that is no longer in the hire fleet and the actor. The use case diagram shows all the details that need to be included for deleting a car from the fleet which are date of registration, registration number, model, make, engine capacity and hire class (1 - 6) that were added by using the include use case tool.

Record a hire and register a new customer
While the use case diagram had to be created the relationship between the actor and the use case had to be identified. When the process was completed the use case diagram could then be created by using the different types of tools such as the actor, use case and association line.

When creating the use cases diagram there were two use cases which had to be involved. ‘Register a new customer’ and ‘Record a hire’ were the two use cases. The reason for this is because the use cases have to rely in each other. This is because the customer has to be a registered customer for them to hire a car.

The use case diagram for ‘Register a new customer’ is associated with an actor. When creating the use case diagrams extend tools and include tools were used to create them. Extend tools were used for ‘Register a new customer’ and ‘Sales assistance ask customer to provide information’. Include tools were used for ‘Sales assistance asks for the customer name, address, telephone number’ and ‘the system saves the new customer’.

The use diagram for ‘Record a hire’ is associated with an actor. When creating the use case diagram include tools were used to create ‘The cars make, engine capacity, model, registration number and hire class’ and ‘The customer’s name, address, telephone number and driving licence’.

Determine if a car is due for a service, and record a service
The 2 uses cases which have been combined are ‘determine if a car is due for a service’ and ‘record a service’. This is because one of these processes is carried out if the previous use case is true, therefore this will show the relationship between the use cases and actors.

For the design the use case called ‘determine if service is due’ was created, which is associated with the actor. Extension points of record a service, this use case was made, which include ‘minor service = 6000 miles’ and ‘major service 12000 miles or above’.

The use case ‘determine if a service is due’ was extended from another use case called ‘record a service’. This use case also had extension points of log a service, which included ‘make and model’, ‘registration number’, ‘date of service’, ‘name of mechanic’ and mileage of time of service’.

Another extended use case called ‘log a service’ is made, is extended to the use case ‘determine if a service is due’.

(*Please continue on to next page…)*

6) Using the use cases and problem description, create a conceptual class diagram showing the classes, associations and attributes that you have identified. Give discussion to support your identification.

Register a new customer

Symbol: Customer
Intention: a person who wants to become a registered customer
Extension: a customer with the name John Smith, Jane Doe etc.

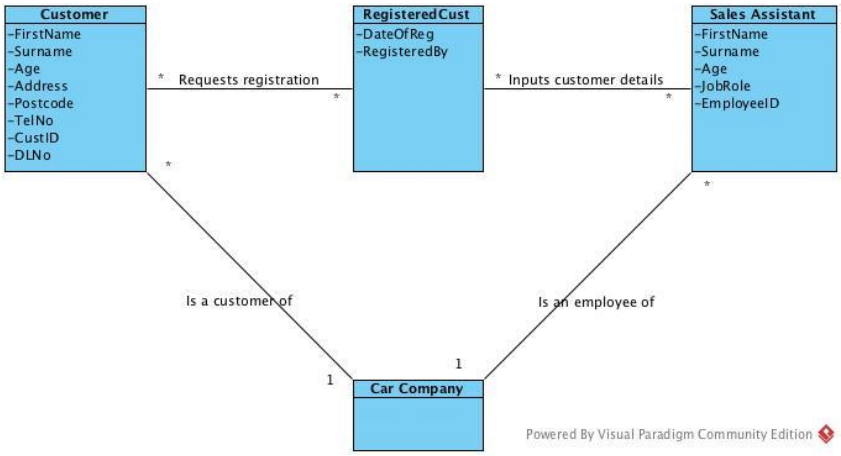
Attributes: FirstName, Surname, Age, Address, Postcode, TelNo, CustID, DLNo

Symbol: Sales assistant
Intention: a person who registers the customer onto the system
Extension: a sales assistant with the name Alan Turner, Mary Reynolds.

Attributes: FirstName, Surname, Age, JobRole, EmployeeID

Symbol: RegisteredCust
Intention: process of registering a customer
Extension: sales assistant Alan Turner registered customer Jane Doe onto the system.

Attributes: DateOfReg, RegisteredBy



Record that a particular car has been hired

Symbol: Customer
Intention: the customer that wants to hire the car
Extension: a customer with the name John Smith, Jane Doe etc.

Attributes: FirstName, Surname, Age, Address, Postcode, TelNo, CustID, DLNo

Symbol: Sales assistant
Intention: a person who uses the system to record a hire
Extension: a sales assistant with the name Alan Turner, Mary Reynolds.

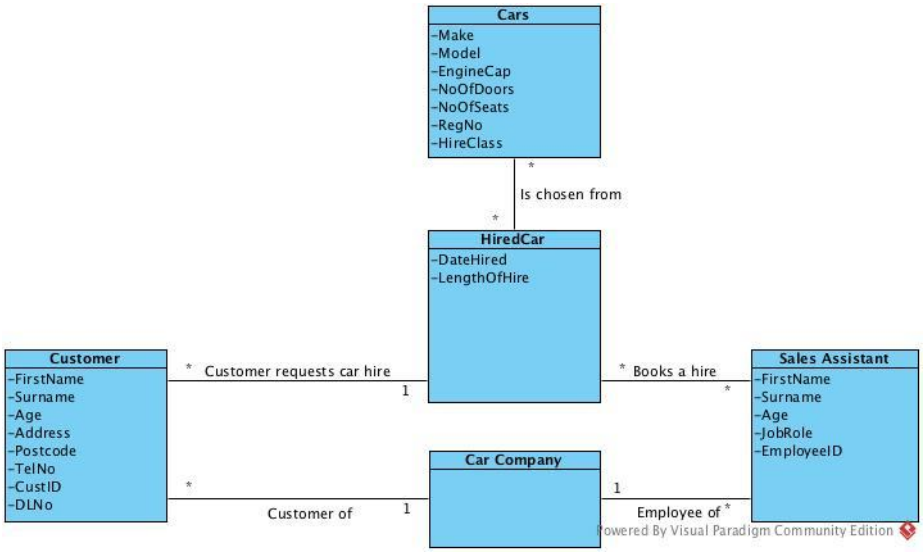
Attributes: FirstName, Surname, Age, JobRole, EmployeeID

Symbol: Cars
Intention: the car that is being hired by a customer
Extension: a car with the make and model like Volkswagen Golf, Volkswagen Polo.

Attributes: Make, Model, EngineCap, NoOfDoors, NoOfSeats, RegNo, HireClass

Symbol: HiredCar
Intention: the process of hiring a car.
Extension: a Volkswagen Polo that has been hired by Jane Doe.

Attributes: DateHired, LengthOfHire



Record that a particular car has been returned

Symbol: Customer
Intention: the customer who is returning the car
Extension: a customer with the name John Smith, Jane Doe etc.

Attributes: FirstName, Surname, Age, Address, Postcode, TelNo, CustID, DLNo

Symbol: Sales assistant
Intention: sales assistant that is recording the hire onto the system
Extension: a sales assistant with the name Alan Turner, Mary Reynolds.

Attributes: FirstName, Surname, Age, JobRole, EmployeeID

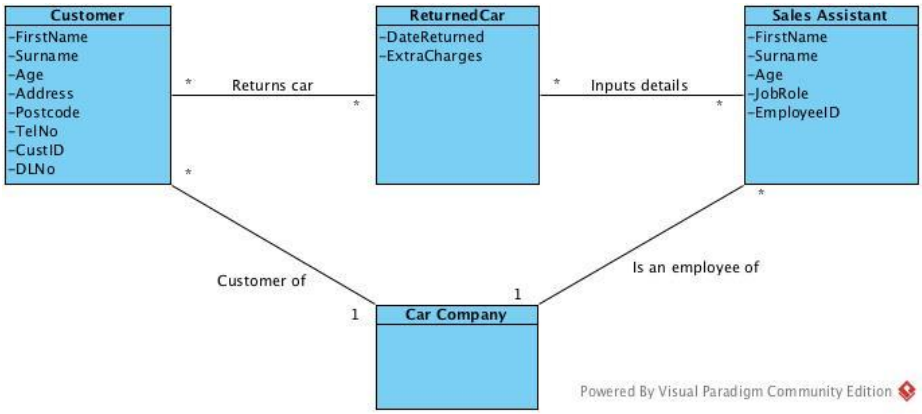
Symbol: Cars
Intention: the car that is being returned by a customer

Extension: a car with the make and model like Volkswagen Golf, Volkswagen Polo.

Attributes: Make, Model, EngineCap, NoOfDoors, NoOfSeats, RegNo, HireClass

Symbol: ReturnedCar
Intention: the process of a customer returning a car
Extension: a customer with the name John Smith returned a Volkswagen Golf

Attributes: DateReturned, ExtraCharges



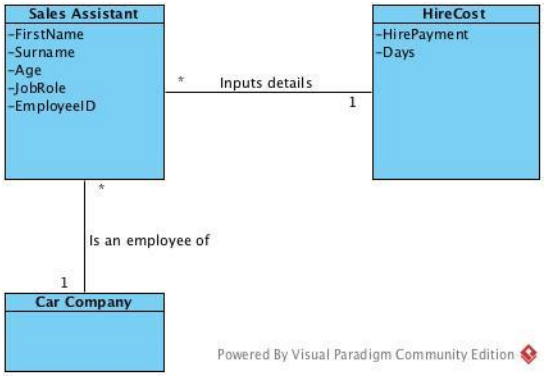
Calculate the cost based on a daily hire rate

Symbol: Sales assistant
Intention: sales assistant that is using the system to retrieve the hire cost
Extension: a sales assistant with the name Alan Turner, Mary Reynolds.

Attributes: FirstName, Surname, Age, JobRole, EmployeeID

Symbol: HireCost
Intention: the event of the system calculating the hire cost of a sale
Extension: hirecost1, hirecost2, hirecost3, etc.

Attributes: HirePayment, Days



Display the appropriate details, and print out a receipt

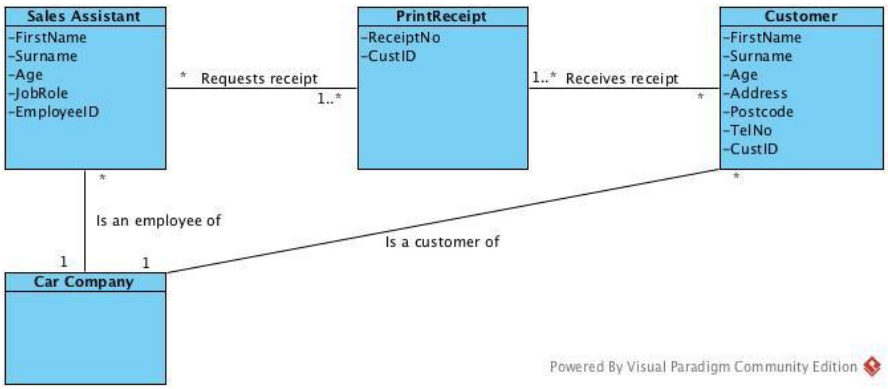
Symbol: Sales assistant
Intention: sales assistant that will tell the system to print a receipt.
Extension: a sales assistant with the name Alan Turner, Mary Reynolds.

Attributes: FirstName, Surname, Age, JobRole, EmployeeID
Symbol: Customer
Intention: the customer that will receive the receipt with their details on.
Extension: a customer with the name John Smith, Jane Doe etc.

Attributes: FirstName, Surname, Age, Address, Postcode, TelNo, CustID

Symbol: PrintReceipt
Intention: event of printing a receipt
Extension: receiptno1, receiptno2, receiptno3.

Attributes: ReceiptNo, CustID,



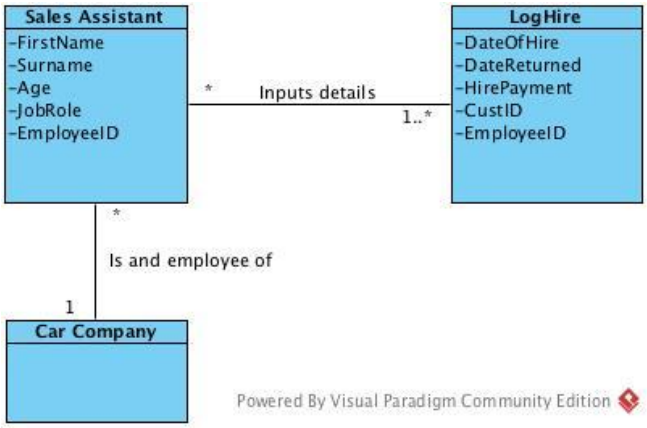
Log a completed hire

Symbol: Sales assistant
Intention: sales assistant that will log the hire onto the system.
Extension: a sales assistant with the name Alan Turner, Mary Reynolds.

Attributes: FirstName, Surname, Age, JobRole, EmployeeID

Symbol: LogHire
Intention: the event of logging a hire onto the system
Extension: a sales assistant Alan Turner logged hire onto system.

Attributes: DateOfHire, DateReturned, HirePayment, CustID, EmployeeID



Determine if a car is ready for a service

Symbol: Sales assistant
Intention: sales assistant that will log the hire onto the system.
Extension: a sales assistant with the name Alan Turner, Mary Reynolds.

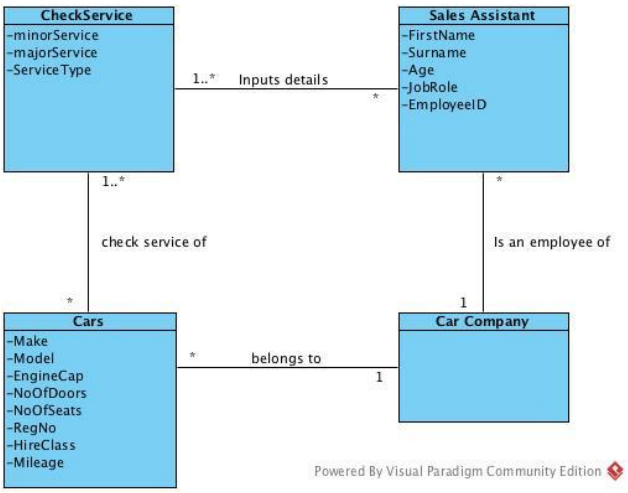
Attributes: FirstName, Surname, Age, JobRole, EmployeeID

Symbol: CheckService
Intention: sales assistant will check the mileage to see if it needs a service.
Extension: service1, service2

Extension: minorService, majorService, ServiceType

Symbol: Cars
Intention: the car that is being returned by a customer
Extension: a car with the make and model like Volkswagen Golf, Volkswagen Polo.

Attributes: Make, Model, EngineCap, NoOfDoors, NoOfSeats, RegNo, HireClass, Mileage



Record a service for a particular car, together with the date of the service, the type of the service and the mechanic responsible for the service

Symbol: Sales assistant
Intention: sales assistant that will log the hire onto the system.
Extension: a sales assistant with the name Alan Turner, Mary Reynolds.

Attributes: FirstName, Surname, Age, JobRole, EmployeeID

Symbol: Cars
Intention: the car that is being recorded onto the system
Extension: a car with the make and model like Volkswagen Golf, Volkswagen Polo.

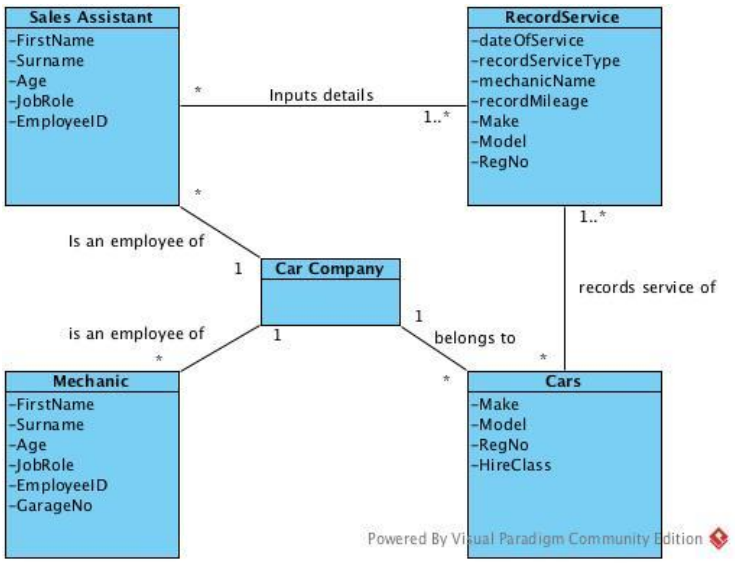
Attributes: Make, Model, RegNo, HireClass

Symbol: Mechanic
Intention: the person who performs a service on a car
Extension: a mechanic with the name Roger James, Bill Allen.

Attributes: FirstName, Surname, Age, JobRole, EmployeeID, GarageNo

Symbol: RecordService
Intention: the even of a service being recorded onto the system
Extension: A sales assistant Alan Turner recorded the service of the Volkswagen Golf onto the system that was performed by Roger James.

Attributes: dateOfService, recordServiceType. mechanicName, recordMileage, make, model, regNo



Add a new car to the fleet

Symbol: Sales assistant
Intention: sales assistant that is adding a new car to the fleet
Extension: a sales assistant with the name Alan Turner, Mary Reynolds.

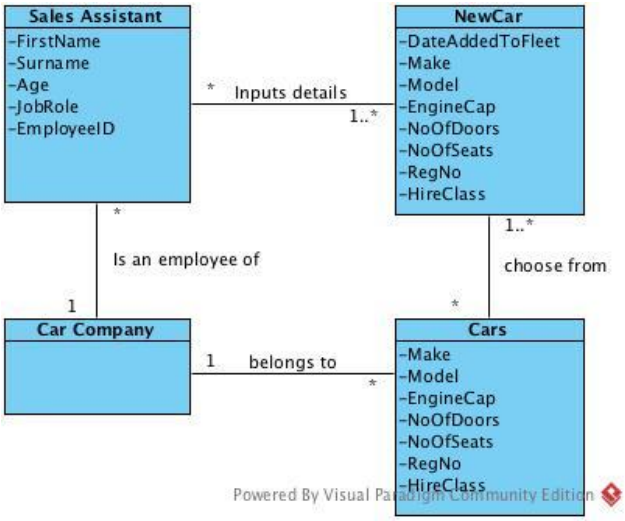
Attributes: FirstName, Surname, Age, JobRole, EmployeeID

Symbol: Car
Intention: the car that is being added into the fleet.
Extension: a car with the make and model like Volkswagen Golf, Volkswagen Polo.

Attributes: Make, Model, EngineCap, NoOfDoors, NoOfSeats, RegNo, HireClass

Symbol: NewCar
Intention: the event of adding a new car to the fleet.
Extension: a sales assistant with the name Mary Reynolds added the car Volkswagen Polo to the fleet.

Attributes: DateAddedToFleet, Make, Model, EngineCap, NoOfDoors, NoOfSeats, RegNo, HireClass



Delete a car that is no longer in the hire fleet

Symbol: Sales assistant
Intention: sales assistant that is adding a new car to the fleet
Extension: a sales assistant with the name Alan Turner, Mary Reynolds.

Attributes: FirstName, Surname, Age, JobRole, EmployeeID

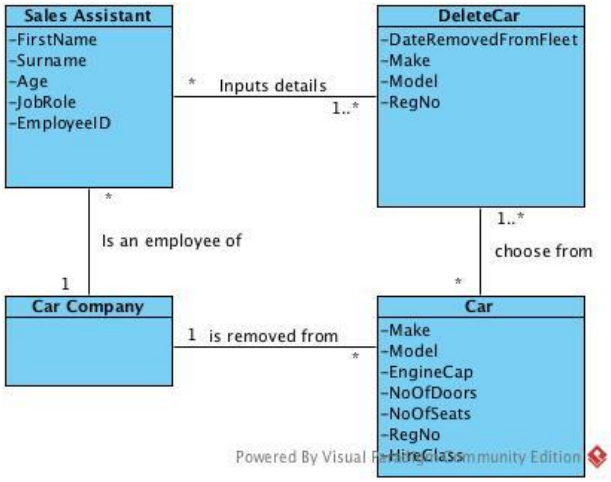
Symbol: Car
Intention: the car that is being deleted from the fleet.

Extension: a car with the make and model like Volkswagen Golf, Volkswagen Polo.

Attributes: Make, Model, EngineCap, NoOfDoors, NoOfSeats, RegNo, HireClass

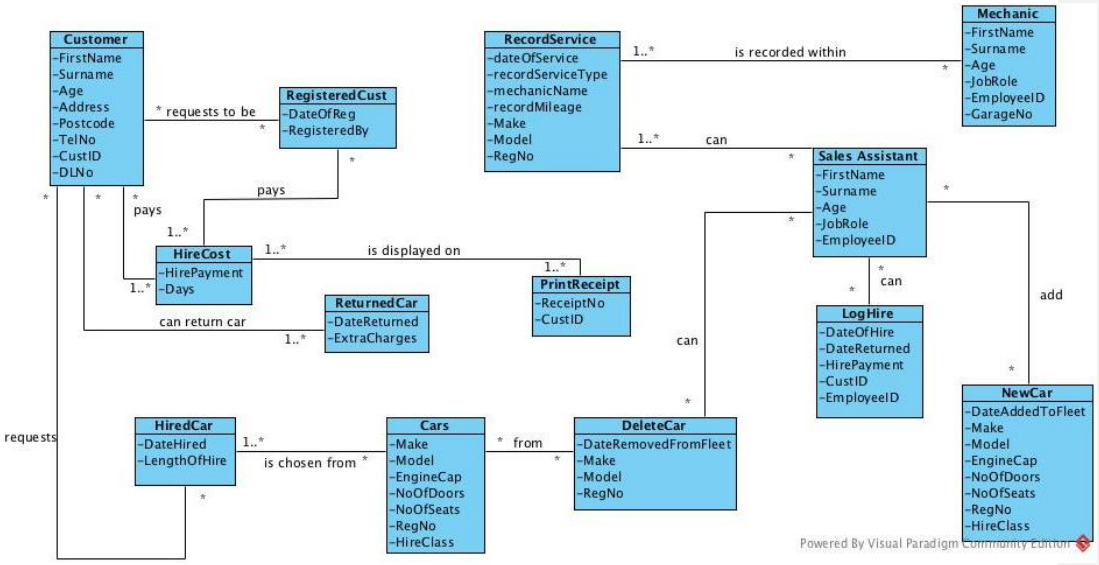
Symbol: DeleteCar
Intention: the event of deleting a car from the fleet.
Extension: a sales assistant with the name Mary Reynolds deleted the car Volkswagen Polo from the fleet.

Attributes: DateRemovedFromFleet, Make, Model, RegNo



(*Please continue to next page…)*

Conceptual Class Diagram

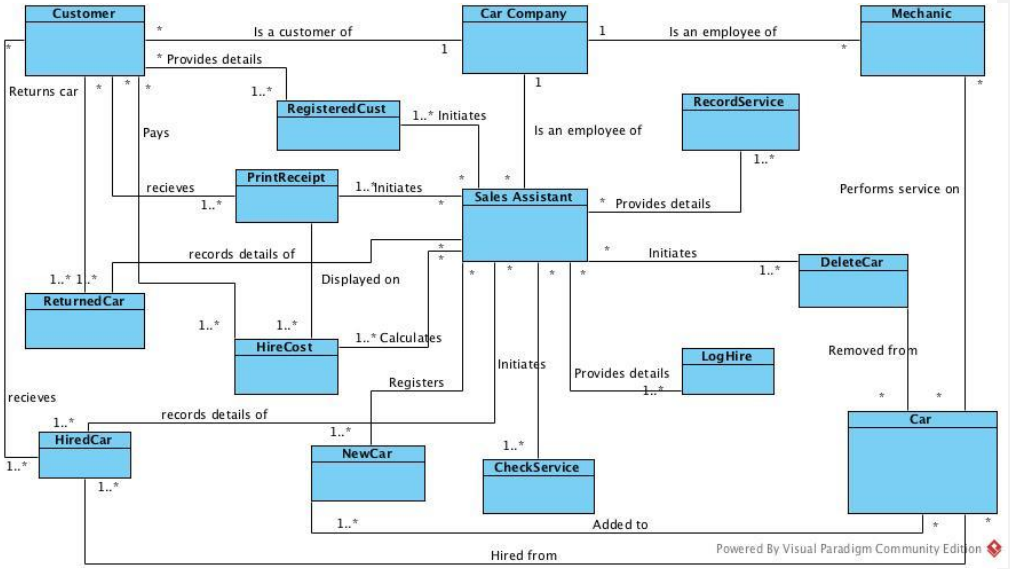


In the conceptual class diagram the classes that were identified as symbols were included. Underneath them are the attributes that will form the data input for the system. Attributes are important as they form the data input for the system. They will need to be inputted by an actor, and they will also form part of the interface of the system. The links and associations are shown, which display how the classes and actors are associated with each other and with different classes.

批注 [ZL7]: Some confusion about conceptual classes and use case operations

(*Please continue to next page…)*

Conceptual model



The conceptual model focuses on the associations between the important concepts and the company. It shows the multiplicity, which focuses on how many possibilities of the class there are. It is similar to the conceptual class model, however it focuses more on the relations between the classes, as opposed to the attributes.

(*Please continue on to next page…)*

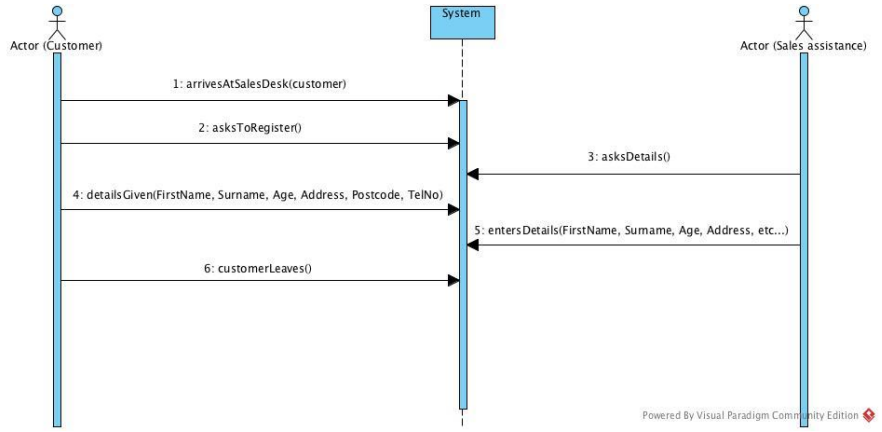
II. Analysis of functionality of system operations

7) Using chapter 6 of the course notes and the techniques used, identify the system operations from the typical course of events of the use cases. Create system sequence diagrams for the use cases you think are the most significant for the development of the system.

Register a customer

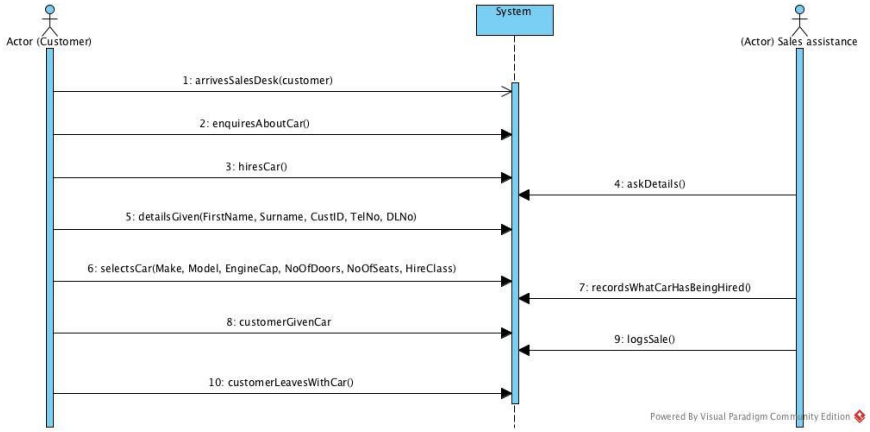
INPUT	OUTPUT
Sales assistant initiates a new customer form	
	System loads new customer form
Sales assistant inputs customer details into the form	
Sales assistant submits the form to the system	
	System receives submitted form
	System checks all entries are valid data types
	System sends confirmation of new customer with customerID
Sales assistant closes customer form	

批注 [ZL8]: Only need to show the direct actor(s), and the interaction between the actor(s) and the system under development



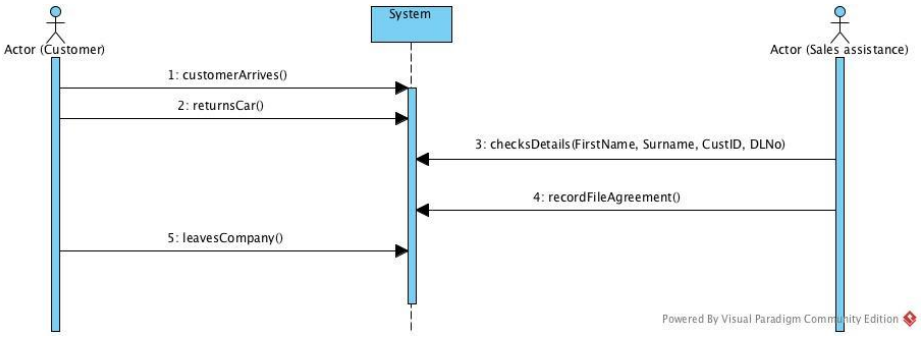
Record a particular car has been hired

INPUT	OUTPUT
Customer selects car, sales assistant enters car selection	
	System records Customer car selection
Sales Assistant collects customer details	
	System assigns customer details with car hire
Sales Assistant completes sale	
	System logs date of sale and details



Record a particular car has been returned

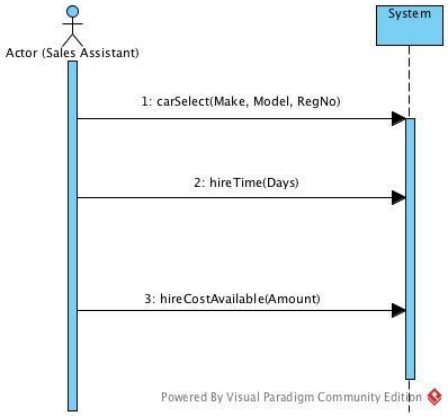
INPUT	OUTPUT
Sales Assistant enters returned cars details	
	System retrieves sale details
	System checks Car details match customer
Sales Assistant takes car and assigns car as returned	
	System assigns Car as returned



(Please continue to the next page…)

Calculate cost based on the daily hire rate

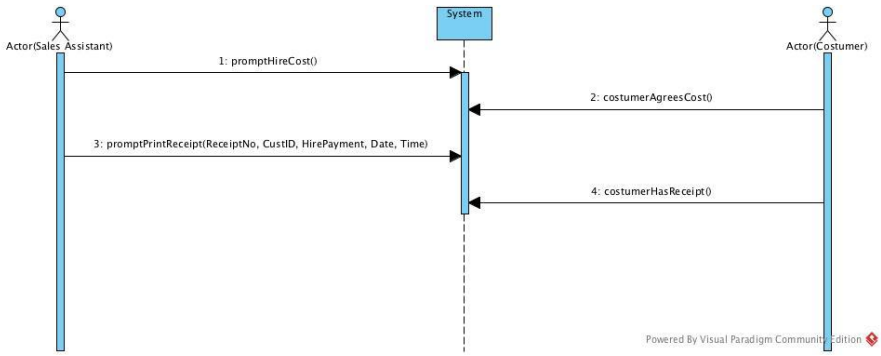
INPUT	OUTPUT
Sales Assistant initiates Car hire form	
	System loads Hire car form
Sales Assistant selects a car	
Sales Assistant adds hire time	
	System receives form
	System ensures data is in correct type
	System calculates hire cost based on customer's specification and returns to Sales assistant the Cost of hire.
Sales Assistant receives cost of hire.	



Display appropriate details and print out a receipt

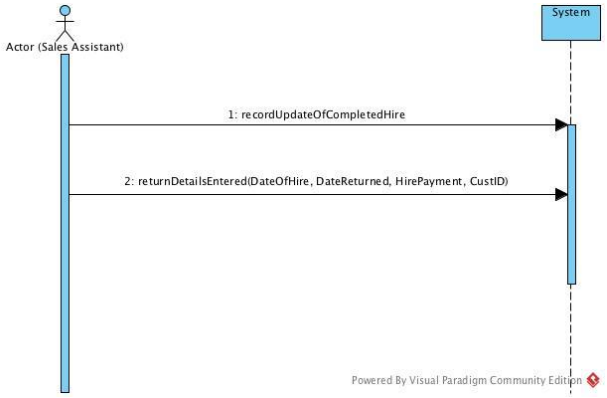
INPUT	OUTPUT
Sales Assistant requests hire cost	
	System returns the hirecost
Customer Agrees cost and payment is made	
	System processes payment
Sales Assistant requests Receipt	
	System gathers necessary details and returns receipt
Sales Assistant gives receipt to customer	

(Please continue to next page…)



Log a completed hire

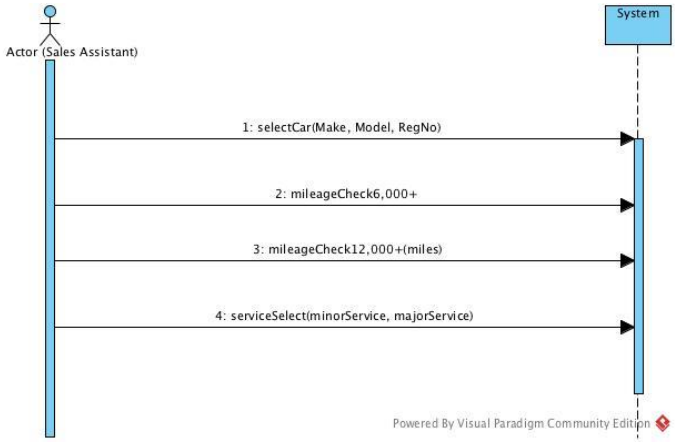
INPUT	OUTPUT
Sales Assistant initiates Completed hire form	
	System returns completed hire form
Sales Assistant enters all necessary details	
	System ensures all are correct data type and valid
	Returns conformation that records have been updated to 'completed'
Sales Assistant receives conformation of completed hire	



(Please continue to next page…)

Determine if a particular car is due for a service

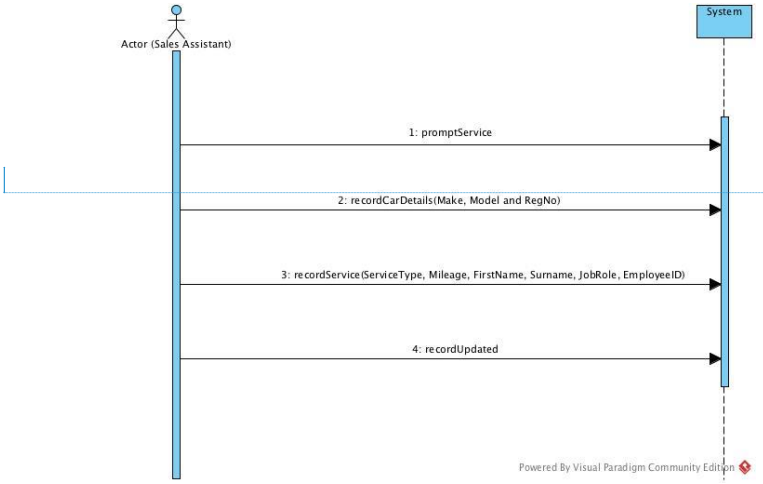
INPUT	OUTPUT
Sales assistant selects car from records and requests a mileage check on the car	
	System finds selected car from records and returns the mileage of the car
Sales Assistant checks whether car has done more than 6000+ miles and 12000+ miles	
Selects Minor or major service	
	System books and notes of necessary service on car records
Sales Assistant receives conformation of necessary service.	



Record a service

INPUT	OUTPUT
Sales Assistant requests Car details	
	System returns Car details
Sales Assistant enters service details	
	System records car service details
	System updates record

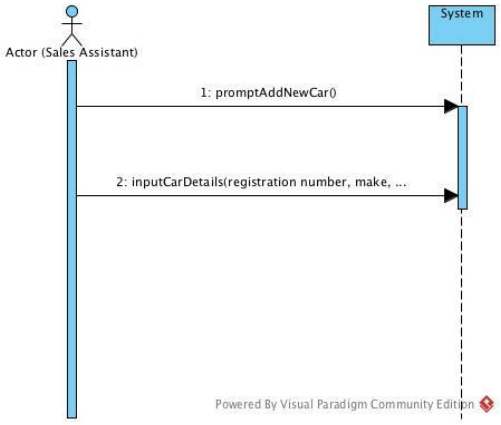
(Please continue to next page…)



批注 **[ZL9]:** How can an actor "prompt" the system?

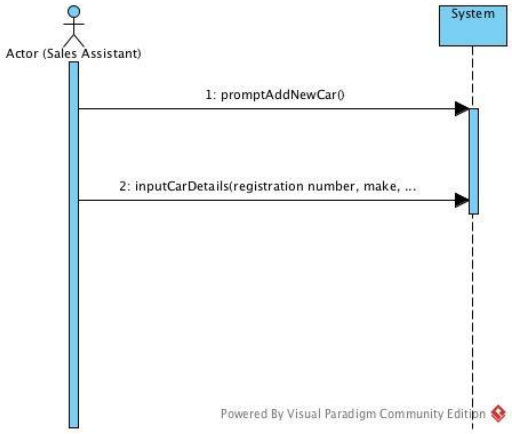
Add a new car to the fleet

INPUT	OUTPUT
Sales Assistant initiates add new car form	
	System loads Add new car form
Sales Assistant inputs car details to new car form	
Sales Assistant submits car form to the system	
	System receives form
	System checks all necessary data has been inputted in correct data type
	System sends conformation of new car being added to the fleet and Assigns carID along with date added to fleet
Sales Assistant closes form.	



Delete a car from the fleet

INPUT	OUTPUT
Sales Assistant initiates Car removal from fleet.	
	System returns Car removal form
Sales Assistant adds Car details for car being removed	
	System validates data inputted
	System asks for conformation of removal
Sales Assistant confirms removal of car	
	System removes the car and returns conformation of removal to Sales Assistant
Sales Assistant receives conformation of removal	



Discussion

System sequence diagrams are part of the requirement analysis phase where you focus on what a system does rather than explain how it's done. The inputs of an actor are focused on within these diagrams and not the output of a system.

From the actor to the system, operations take place but no output is shown and from the 2nd actor to the system, more operations take place. None of the outputs for any operations are shown.

(*Please continue to next page…)*

8) Based on your use case model and conceptual model that you have produced write the contracts for the system operations that you have identified.

Start up

Name: StartSystem()
Responsibilities: Start the system ready for data input
Post-conditions:

- *SalesDesk, CarCatalog and FuntionList* is created.

批注 [ZL10]: ?

批注 [ZL11]: was

Register a new customer

Name: arrivesAtSalesDesk()
Responsibilities: The customer arrives at the sales desk
Type: Concept
Cross-reference: Use cases: Register a new customer
Post-conditions: promptRegistration() contract begins.

Name: promptRegistration()
Responsibilities: sales assistant prompts the registration of a customer.
Type: Interface
Cross-reference: Use cases: Register a new customer
Pre-conditions: arrivesAtSalesDesk() has been completed

批注 [ZL12]: ?

Post-conditions:

- detailsGiven() contract starts
- FunctionList has been accessed

Name: entersDetails(FirstName, Surname, Age, Address, Postcode, TelNo)
Responsibilities: The sales assistances enters the details to the system
Type: System
Cross reference: Use case: Register a new customer 5
Output: The sales assistance enters the details of the customer into the system
Pre-conditions: detailsGiven(name, address, telephone number) has been completed

- Post-condition: *NewCust* is created
- The customerLeaves() contract begins

Name: customerLeaves()
Responsibilities: The customer leaves company
Type:
Concept
Cross reference: Use case: Register a new customer 6
Output: The customer leaves or enquiries about something else
Pre-conditions: entersDetails() has been completed

Record that a particular car has been hired

Name: arrivesSalesDesk()
Responsibilities: The customer arrives at the sales desk
Type: Concept
Cross reference: Use case: Record that a particular car has been hired 1
Post-conditions: Customer begins requestCarHire() contract

Name: requestsCarHire()
Responsibilities: The customer request they would want to hire a car
Type: Concept

Cross reference: Use case: Record that a particular car has been hired 2
Pre-condition: arrivesSalesDesk(Customer) has been completed
Post-condition: The sales assistance begins askDetails() contract

Name: asksDetails()
Responsibilities: The sales assistance asks the customers for their details
Type: Concept
Cross reference:
Use case: Record that a particular car has been hired 3
Pre-condition: requestsCarHire() has been completed
Post-condition: The customer begins detailGiven() contract

Name: detailsGiven(FirstName, Surname, CustID, TelNo, DLNo)
Responsibilities: The customer gives their details to the sales assistance
Type: Interface
Cross reference:
Use case: Record that a particular car has been hired 4
Exceptions: The sales assistance misspelled the customers name or telephone number
Pre-condition: askDetails() has been completed
Post-condition:

- The customer begins selectsCar() contract
- *NewCarHire* is created.

Name: selectsCar(Make, Model, EngineCap, NoOfDoor, NoOfSeats, HireClass)
Responsibilities: The customer selects what car they refer
Type: Interface
Cross reference:
Use case: Record that a particular car has been hired 5
Exception: The make and engine capacity are invalid or they don't exits
Pre-conditions: detailsGiven() has been completed, all inputs valid.
Post-conditions: Begins recordsWhatCarHasBeenHired() contract.

Name: recordsWhatCarHasBeenHired()
Responsibilities: The sales assistance records what car has been hired
Type: System
Cross reference: Use case: Record that a particular car has been hired 6
Exception: The righty car was not recorded or the sales assistance forgot to record what car had been hired
Pre-condition: selectsCar() has been competed
Post-condition: Begins customerLeavesWithCar() contract

Name: customerLeavesWithCar()
Responsibilities: The customer leaves with the car they have hired
Type: Concept
Cross reference: Use case: Record that a particular car has been hired 7
Output: The customer leaves or enquiries about something else
Pre-condition: recordsWhatCarHasBeenHired() has been completed

Record that a particular car has been returned

Name: customerArrives()
Responsibilities: The customer arrives at the sales desk
Type: Concept
Cross reference:
Use case: Record that a particular car has been retuned 1
Post-conditions: begin returnsCar() contract

Name: returnsCar()
Responsibilities: The customer returns the car they have hired
Type: System
Cross reference: Use case: Record that a particular car has been returned 2
Post-conditions: begin checksDetails() contract

Name: checksDetails(FirstName, Surname, CustID, DLNo)
Responsibilities: The sales assistance checks the details of the customer to see if they have retuned the correct car they had hired
Type: System
Cross reference:
Use case: Record that a particular car has been returned.
Pre-conditions: all details are correct. If not an error warning is issued.
Post-conditions:

- The system begins contract recordFileAgreement()
- *ReturnedCarHire* created.

Name: recordFileAgreement()
Responsibilities: The sales assistance records the evidence into the hire file agreement to state the car was returned
Type: System
Cross reference:
Use case: Record that a particular car has been hired 4
Exception: The file agreement was misplaced or was lost
Pre-condition: checksDetails() has been competed
Post-condition: begin leavesCompany() contract

Name: leavesCompany()
Responsibilities: The customer leaves the company after returning the car
Type: Concept
Cross reference:
Use case: Record that a particular car has been returned 5
Output: The customer leaves or enquiries about something else
Pre-condition: recordFileAgreement() has been complete

Calculate the cost based on the daily hire rate

Name: carSelect(Make, Model, RegNo)
Responsibilities: Select a specific car and bring up all of its details
Type: System
Cross References: Use Case: Calculate the cost based on the daily hire rate
Exceptions: If registration is not recognised then prompt the user with an error stating the reason for the error.
Pre-conditions: Registration number known to the system
Post-conditions:

- The system pulls up the cars details relating to the registration number.
- *NewCarSelect* is created.

Name: hireTime(Days)
Responsibilities: Check the number of days the car has been hired out for
Type: System
Cross References: Use Case: Calculate the cost based on the daily hire rate
Exceptions: If a time period has not been selected then prompt the user to fill in the hire start and hire finish fields.
Output:

Pre-conditions: The duration of the car hire per day known to the system
Post-conditions: The system calculates the total hire time

Name: costCalculation()
Responsibilities: Calculate the cost of the total hire by the cost per day *(times) the number of days
Type: System
Cross References: Use Case: Calculate the cost based on the daily hire rate
Exceptions: If there is no hire duration or cost per day then prompts the user about this error
Pre-conditions: The daily cost and the hire time should be known to the system
Post-conditions: The system calculates the total cost of the hire

Name: hireCostAvailable(HireCost)
Responsibilities: The hire cost is correctly displayed on the system
Type: System
Cross References: Use Case: Calculate the cost based on the daily hire rate
Exceptions: if no cost is available then prompt the user of this error
Pre-conditions: The calculations already calculated
Post-conditions: The systems displays the total cost for future reference

Display the appropriate details, and print out a receipt

Name: promptHireCost()
Responsibilities: Sales assistant prompts system for costumers hire cost.
Type: System
Cross References: Use case: Display the appropriate details, and print out a receipt.
Post-conditions: System displays hire cost

Name: promptPrintReceipt(ReceiptNo, CustID, HirePayment, Date, Time)
Responsibilities: Sales assistant prompts system to print customer's receipt.
Type: System
Cross References: Use case: Display the appropriate details, and print out a receipt.
Pre-conditions: promptHireCost() was completed
Post-conditions:

- A receipt was printed.
- *NewReceipt* was created.

Name: costumerAquiresReceipt()
Responsibilities: Costumer acquires there receipt from the sales assistant.
Type: Concept
Cross References:
Use case: Display the appropriate details, and print out a receipt.
Note:
Exceptions:
Output:
Pre-conditions: promptPrintReceipt() was completed
Post-conditions: Costumer leaves with receipt

Log a completed hire

Name: recordUpdateOfCompletedHire(RegNo)
Responsibilities: keep an up-to-date record of all completed hires
Type: System
Cross References: Use Case: Log a completed hire
Note:
Exceptions: Prompt the user if there is no record of a hire when the registration is entered

Output:
Pre-conditions: The car must be returned prior a record update
Post-conditions: The system will be updated

Name: returnDateEntered(DateReturned)
Responsibilities: The date of the return must be entered to indicate the end of hire
Type: System
Cross References: Use Case: Log a completed hire
Exceptions: The date must be after the start date otherwise an error message will occur
Pre-conditions: The car must be present for a date to be entered
Post-conditions: Date of the return is now entered and the record is updated

Name: amountToPay(HirePayment)
Responsibilities: Display the cost of the hire
Type: System
Cross References: Use Case: Log a completed hire
Exceptions: If no cost is available then system notifies the user
Pre-conditions: Cost per day and the number of days hired are already known to the system
Post-conditions: Daily cost and duration are calculated to show the amount to pay

Record a service for a particular car, together with the date of the service, the type of the service, and the name of the mechanic responsible

Name: promptCarDetails()
Responsibilities: prompts the user to input car details
Type: System
Cross References: Use Case: Record a service for a particular car
Post-conditions: the user inputs car details recordCarDetails()

Name: recordCarDetails(Make, Model, RegNo)
Responsibilities: inputs car details to find out car information
Type: Interface
Cross References: use case: record a service for a particular car
Exceptions: If there is no car that satisfies the input details then the system will display an error.
Post-conditions: system shows car details with mileage.

Name: recordService(ServiceType, Mileage, FirstName, Surname, JobRole, EmployeeID)
Responsibilities: the user records or books a service for the car if it is needed.
Type: Interface
Cross References: use case: Record a service for a particular car
Exceptions: a service is not needed and no record will be updated.
Pre-conditions: the car details entered are all valid.
Post-Conditions:

- system updates car record with a service.
- *NewServiceRecord* is created.

Add a new car to the fleet

Name: promptAddNewCar()
Responsibilities: Sales assistant prompts system to add a new car.
Type: System
Cross References: Use case: Add a new car to the fleet
Post-conditions: Sales assistant moves on to inputCarDetails()

Name: inputCarDetails(Make, Model, EngineCap, NoOfDoors, NoOfSeats, RegNo, HireClass)
Responsibilities: Sales assistant inputs the details of the new car into the system.

Type: System
Cross References: Use case: Add a new car to the fleet
Pre-conditions: car details was valid
Post-conditions:

- A new car was added to system
- *NewCarToFleet* is created.

Delete a car that is no longer in the hire fleet

Name: inputCarDetails(Make, Model, RegNo)
Responsibilities: Sales assistant inputs the details of the car to be deleted into the system
Type: System
Cross References: Use case: Delete a car that is no longer in the hire fleet.
Pre-conditions: car details were valid
Post-conditions:

- Sales assistant moves on to promptDeleteCar()
- *CarDeletedFromFleet* is created.

Name: promptDeleteCar()
Responsibilities: Sales assistant prompts the system to delete the car.
Type: System
Cross References: Use case: Delete a car that is no longer in the hire fleet.
Pre-conditions: inputCarDetails() was correct.
Post-conditions: Car was deleted form system

Determine if a particular car is due for a particular service

Name: selectCar(Make, Model, RegNo)
Responsibilities: Select a specific car and bring up all of its details
Type: System
Cross References: Use Case: Determine if a particular car is due for a particular service
Exceptions: If registration is not recognised then prompt the user with an error stating the reason for the error
Pre-conditions: Registration number known to the system
Post-conditions: the system pulls up the cars details relating to the registration number

Name: mileageCheck6,000+(miles)
Responsibilities: Mileage check of 6,000+ to determine if a minor service is needed Type: System
Cross References: Use Case: Determine if a particular car is due for a particular service
Exceptions: mileage record may not be up to date
Pre-conditions: current mileage of the car is known to the system
Post-conditions: Appropriate service has been assigned to the car

Name: mileageCheck12,000+(miles)
Responsibilities: Mileage check of 12,000+ to determine if a major service is needed
Type: System
Cross References: Use Case: Determine if a particular car is due for a particular service
Exceptions: mileage record may not be up to date
Pre-conditions: current mileage of the car is known to the system
Post-conditions: Appropriate service has been assigned to the car

Name: serviceSelect(minorService, majorService)
Responsibilities:
Type: System
Cross References: Use Case: Determine if a particular car is due for a particular service
Exceptions: none of the services may be required

Pre-conditions: 3 options are available: no service, minor service or major service
Post-conditions: Appropriate service has been assigned to the car

Discussion

Contracts are used to describe a functionality of an operation. The operations are selected from the system sequence diagrams and then each system operation was then made into contracts.

The contracts include a number of headers, which are important in their own way. I will include and explain each header that is included in the contracts:

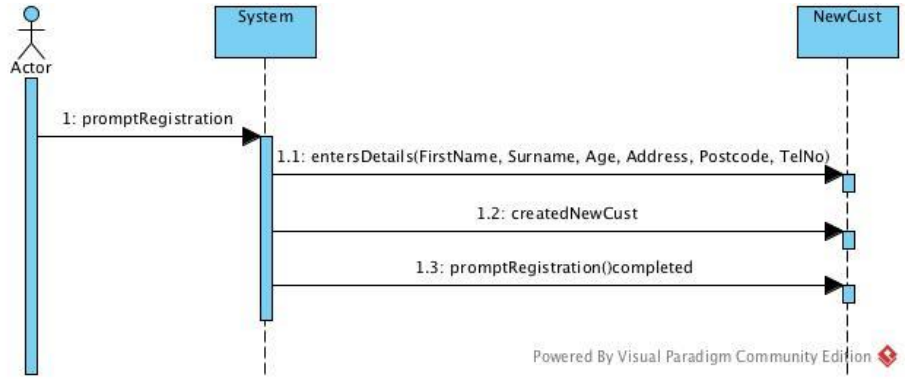
Name: Name of operation from the system sequence diagram
Responsibilities: The reason for this contract and its purpose
Type: a choice between concept, software class, interface or system
Cross References: The name of the use case where the operation is from
Note: Design notes, algorithms, and so on.
Exceptions: Exceptional cases.
Output: Non-UI outputs, such as messages or records that are sent outside of the system.
Pre-conditions: The conditions before an execution takes place
Post-conditions: The conditions when an execution has finished

批注 **[ZL13]:** Serious work, but showing misunderstanding of concepts and principles

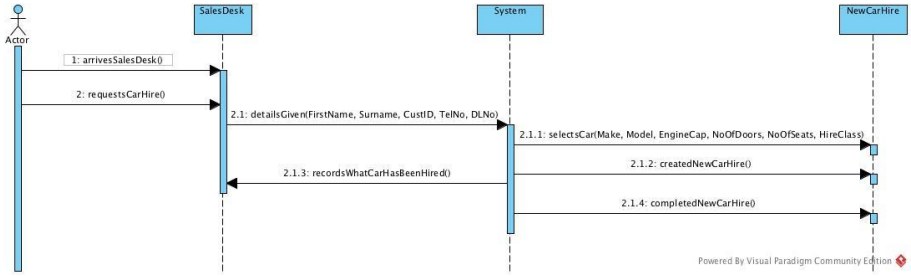
(*Please continue to next page…)*

9) Collaboration diagrams or object sequence diagrams which show the assignment of responsibilities to classes of objects.

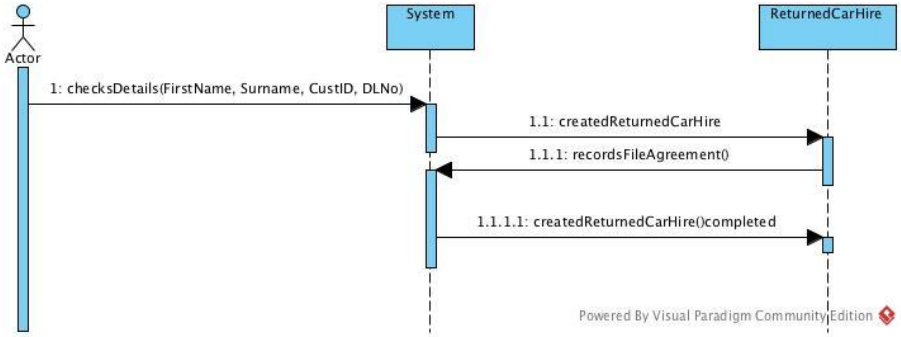
Register a new customer



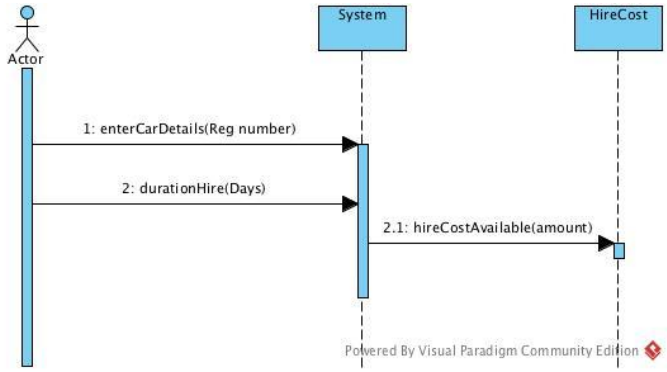
Record that a particular car has been hired



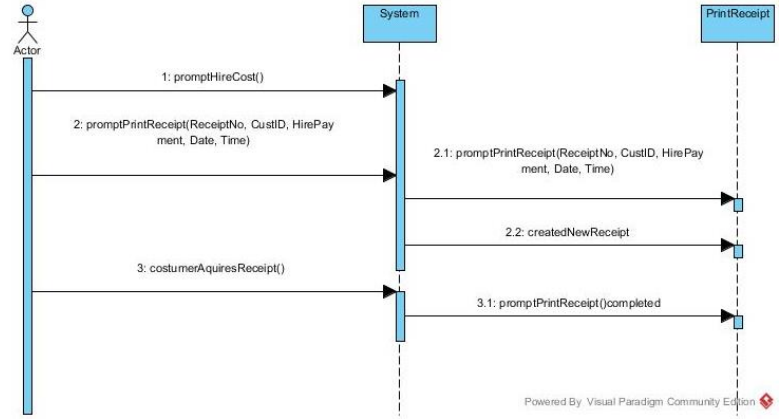
Record that a particular car has been returned



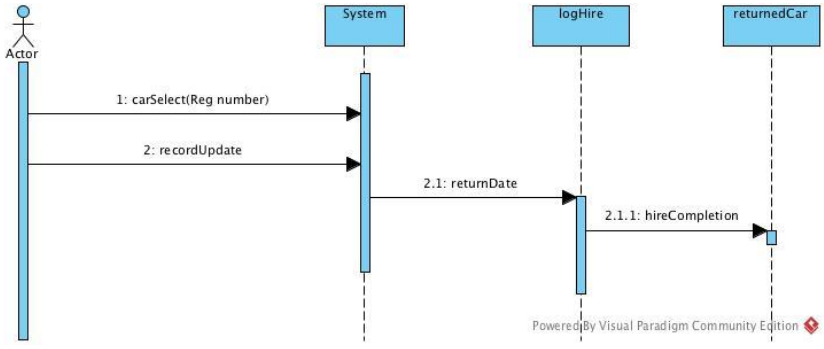
Calculate the cost based on the daily hire rate



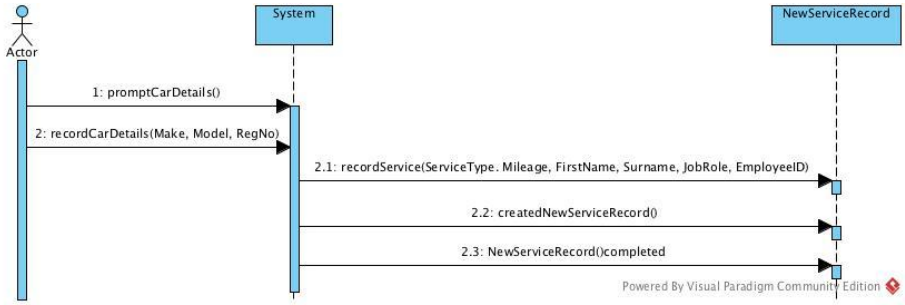
Display the appropriate details, and print out a receipt



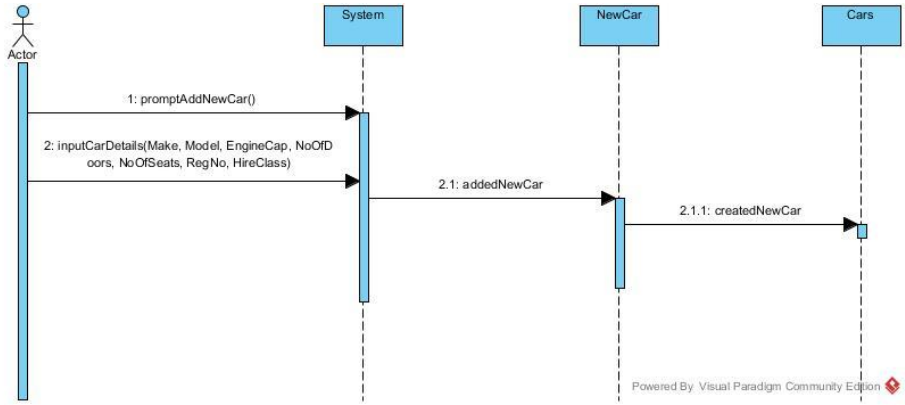
Log a completed hire



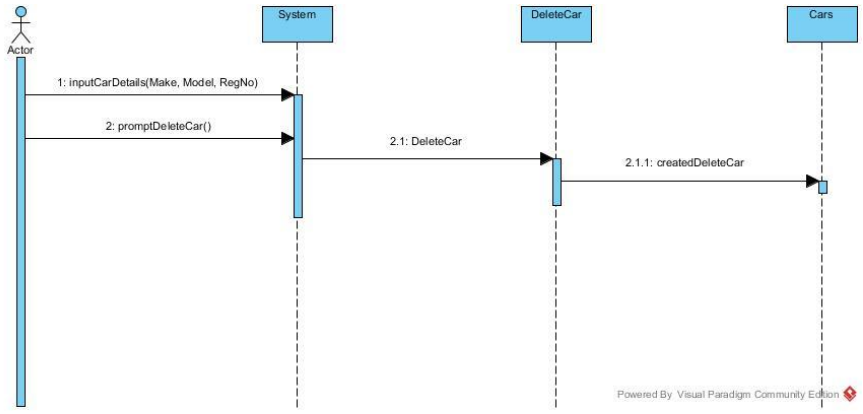
Record a service for a particular car, together with the date of the service, the type of the service, and the name of the mechanic responsible



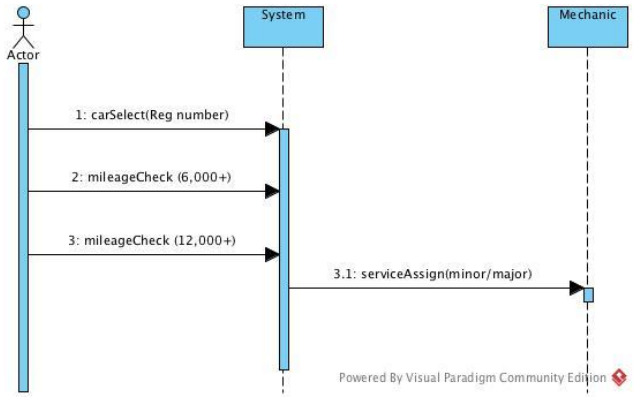
Add a new car to the fleet



Delete a car that is no longer in the hire fleet



Determine if a particular car is due for a particular service



Discussion

An object sequence diagram is focused on the interaction between objects within the system. The conceptual and the use case model are both needed when the object sequence diagrams are created, pre and post conditions are important when creating this diagram

批注 [ZL14]:
Serious work, but showing misunderstanding of concepts and principles

(*Please continue to next page…)*
10) Discuss the use of patterns in your assignment of responsibilities to classes of objects.

Creator Pattern
Pattern Name: Creator
Solution: Assigning the responsibility to create an instance of a class.

- *RequestsCarHire* has attributes "*detailsGiven*" and "*selectsCar*"
- Meaning that *requestCarHire* needs input values for these two attributes
- Hence the message *carHireComplete* from *SalesDesk* to the *System* should consist of "details" and "*CarInformation*"
- Once car hire is successful *SalesDesk* passes them in the creation message to the *System*

Problem: Who/what should be responsible for knowing *CarInformation?*

- The actor at the *SalesDesk* should have the knowledge to provide *Make,Model,EngineCap, NoOfDoors, NoOfSeats, HireClass* of each to be hired car.

Controller Pattern
Pattern Name: Controller
Solution: Assigning the responsibility for handling an input to a class.

- *RecordCarDetails* has an attribute "recordService"
- Meaning that *recordCarDetails* needs input values for that attribute
- Thus the message *NewServiceRecordComplete* from *SalesDesk* to the *System* should consist of *"serviceInformation"*
- Once the "*recordService"* is completed at the *SalesDesk* it is the passed in the creation message to the *System*

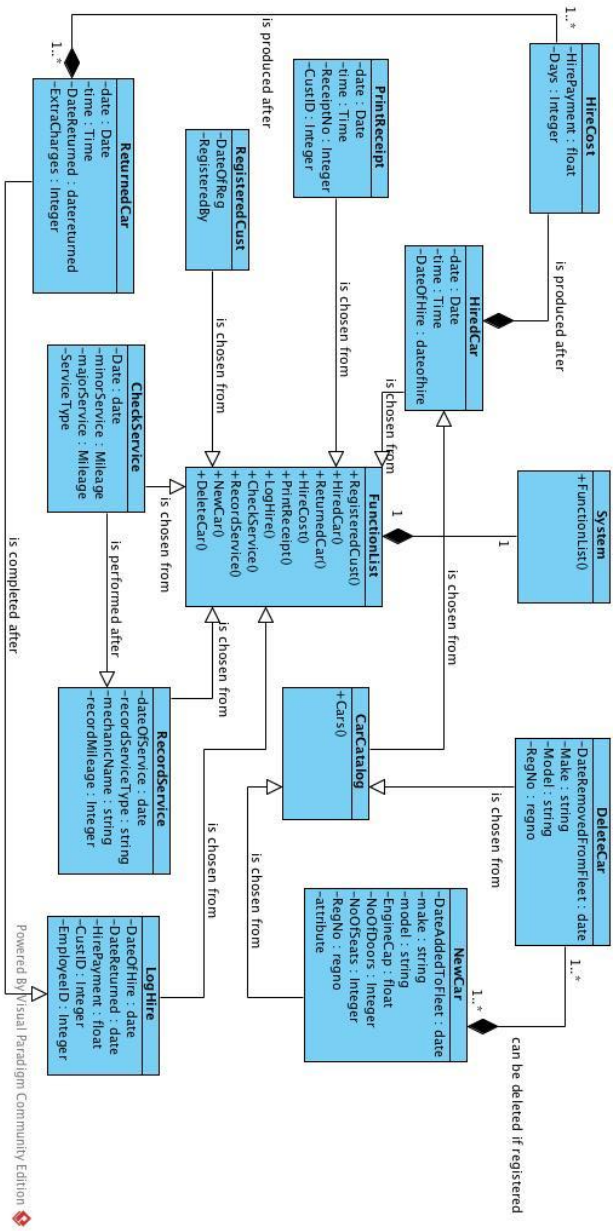
Problem: Who should be responsible for handling an external input event?

- The actor at the *SalesDesk* will be responsible for gathering the significant information from the car "ServiceType" and "Mileage".

The above creator and controller patterns focus of the main functions of the system. They also contain problem-solving methods that revolve around inputting the correct data types and then displaying error messages when the incorrect data type is inputted. The creator pattern ensures that all of the concepts and classes have been assigned the correct attributes. If the correct attributes are not assigned then it means that there will be invalid or incomplete data. They also need to be correct to move throughout the system. For example: if one department or sub-system wanted to view data inputted by another sub-system then they would be able to do so as the data has been correctly inputted.

(Please continue to next page…)
11) The design class diagrams, which shows the methods/operations of classes.

Design class diagram



Discussion

For the design class diagrams it is important to show the associations between the classes. However, actors such as sales assistant, mechanic and customer weren't included, as they are not needed. In the diagram it is important to show the different associations, where two classes are connected, or where one is dependent upon another class. It also includes the appropriate attributes where stated, paired with the data type, such as: string, float or integer.

(*Please continues on to next page…)*
Meeting attendance

Present	
Absent	

Names	**Lesson Attendance (Week commencing)**								
	20th January 2015	**27th January 2015**	**3rd February 2015**	**10th February 2015**	**17th February 2015**	**24th February 2015**	**3rd March 2015**	**15th April 2015**	**21st April 2015**
Imaad Yasin									
Jamie Uddin									
Lucy Shone									
Mohammed Yaseen									
Aleksy Ziarniecki									
Bradley Smith									
Gurpreet Singh									

Names	**Project Meetings Attendance**					
	Meeting 1	Meeting 2	Meeting 3	Meeting 4	Meeting 5	Meeting 6
Imaad Yasin						
Jamie Uddin						
Lucy Shone						
Mohammed Yaseen						
Aleksy Ziarniecki						
Bradley Smith						
Gurpreet Singh						

Task management and completion

TASK	COMPLETED BY
1.1	Gurpreet
1.2	Lucy, Brad
1.3	Jamie, Imaad, Yaseen
1.4	Jamie, Imaad, Yaseen, Lucy
1.5	Jamie, Imaad, Yaseen, Lucy, Alex, Brad
1.6	Lucy
2.1	Jamie, Imaad, Yaseen, Brad
2.2	Jamie, Imaad, Yaseen
3.1	Jamie, Yaseen, Brad
3.2	Alex, Brad
3.3	Lucy
Project management	Imaad, Lucy
Glossary	Imaad
Documentation	Lucy

(*Please continue on to next page…)*
Glossary

Attributes – Attributes are properties or characteristics, an example customer is an attribute of Age

Classes – A group of attributes e.g. Student with the attributes: FirstName, Surname etc.

Conceptual model - A model to show relations between classes and attributes.

Cross-reference – This is related to the use case and the function systems

Contracts – Creating contracts to show what the systems operations do

Essential use cases – The important use cases

Exceptions – A case where a different output will be expected

Intention – What the person intents to do

Integers – Data type that stores whole numbers.

Objects – An object is where data are stored in

Pre-conditions – A condition which happens before an event

Post-conditions – A condition which happens after the event

Strings – Data type that stores alphabetical letters e.g. Name.

Use cases – A list of steps which can be performed by an actor a system