1.1  
The application of a systematic, disciplined, **Quantifiable** approach to development, operation and maintenance of software. That is the approach of engineering to software.

1.2  
CBSE: The approach of reusing components to develop software.  
eg List Car   
  
Decreased Lead Time: The reusing of Search for Cars will decrease development time.  
  
Increased ROI: The resuing or buying of components (List Car) will decrease the development time, allowing cars to be listed sooner, making sales sooner therefore making more money in the end.

**Maintenance, higher quality, spread out costs (leverage costs)**  
  
1.3  
Control

How is control managed? Assign sales rep cannot occur before List Car has been initiated.  
  
How is control Transferred? We can see that once List Car occurs, it transfers control to Assign Sales Rep.

**Hierarchy: which controls have most control NBNBNB**

Data:  
**Data Communication**:   
We can see that once a car is added, that data is stored in the database. This data can be retrieved by searching for a car in the database.  
  
Data Flow: Once the Add car component occurs and the car data is publicly available for sale on the website, buyers can the use search for Cars to retrieve this information, then selecte private viewing, and then submit an offer to purchase. The Seller can either accept the request or reject it. If it is rejected, nothing changes, allowing other buyers to go through the process again. This shows the cyclic flow of data in the system.

Passive or active data: We can see that the system uses active data, as as soon as a car data is added to the system, that car data is viewable from the search for cars component.  
  
Q2  
  
(3) descrir aim of level  
(1) consideration  
  
**Data**

Data vs Information: At this level we look at the difference between data and information. Data is raw facts, while information is data that has been transformed/manipulated into something useful the user can view/understand.

Data Continuity: WE look at what data is put into the system, how this data is transformed into information, what data is being outputted, and the communication of data in the system (between components). WE can also look at how this process can be improved, and what data the system needs to grow/evolve.  
  
How is data represented: Should the data be represented directly, or indirectly? It is important for users to be able to easily understand any displayed data ie no tables.

**Considerations**

When a user if browsing the website through many cars, you only want to show the user the most useful information to try and entice them (image, name, model). You can then add more information on the car if the user clicks on it ie Milage, Service years ect.

**Architecture \*\*\***

Defining the patterns used to achieve the requirements for the specific domain.  
  
Defining the softwares major structural elements and their relationships.  
  
Defining the core level view: so as to define the classes and their associations/how they communicate with eachotehr.

**Consideratios**

When deciding on the architecrural style for the system, one important thing to note is if the data is passive or active (must be active). In the system, we can see that components require up to date information, as you cant buy a car that has already been sold. This could result in loss of customers. Therefore a active data style must be used.

**Interface**

Designing the overall appearance and design of the system.  
  
Defining the flow of information in, between and out of the system.

Defining the interface for:  
User Interface: The interface that users use to interact with the system.  
Internal Interface: How components communicate with eachother   
External Interface: How components communicate with the DB

It is important to note that appearance is subjective, and many different users will be using the app. It is also important to make the UI user friendly andeasy to use to keep users on the platform.   
  
**Considerations**

Therefore a consideration that should be made is having a User Notifications/Messages Interface, so that if a user has many cars up for sale, or many offers for a singular car, they can view them all to make their decision (instead of eg being a one time message/offer).

**Component**

Transforms structural elements form the architecture, into procedural descricption of software components.  
  
Define the internal component details. What they will store, their relationships. And which ones we will have.

We use information form the class based and behaviroal models to design the componenet design.

**Consideration**

If we are to look at the Search for cars component, this component is used on their website. It must also be able to send and retrieve information from the database using HTTPS over IP/TCP.

Q3.1  
  
Separation of Concers: This looks at breaking learger problems (The act of purchasing a car), into smaller, more manageable problems (submit an offer, search for cars, select private viewing of car). The goal of this is to make a once unachievable problem, achievable. It is also good for simplifying a proble, making it both easier and faster to complete. The modularity of this is also good for any updates/maintanance that need to take place. As for any bug fixes of eg Search for Cars, you only have to look at that component to find the problem, instead of the entire Purchase susbsyte.

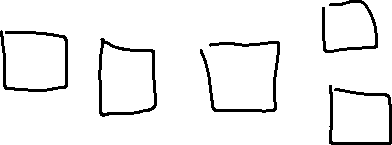
Abstraction

This is the act of hiding unnecessary information from the user. This is to make both the user experience easier nad simpler to use. This allows users to make use of functions that they may not be able to understand. Ie The sales rep just needs to know how to input milage, car info and then click Add Car. They don’t need to know how that milage is then calculated to determine patterns, or how that function communicates and sends data to the DB, for the info to be stored.  
  
Procedural: This is the flow of operations of a function. – eg search for cars – click search button, retrieve cars info from DB, display Cars to user.   
  
Data: The group of data that describes the objects that are stored. Eg User – has a username, password ect

3.2

Data Flow Architecture:   
Components are called in a sequenctial pattern, processing data from x to Y (mention these steps from text)  
You can go directly from design to implementation phase, saving a large amount of time (time = money, sell more cars)  
As components sequential, has active/most up to data info

3.3  
Data Flow architecture.



Components have an expected data input.  
Components can output data to the following component in the sequence.  
Components cannot see nheiboring components details.  
  
This model is a sequence of components connected that run one after another, transforming/manipulating data through multiple steps.  
  
A disadvantage of this diagram is if one goes down, they al do.  
Advantage: You can easily go from the to the design to implementation hase.

Look at Q2 again

Q1.1

The application of a sequential disciplinced, quantificable approach to development, operation and maintenance of software; that is the application of engineering ot software.

Q1.2  
CBSE: The approach of reusing components as a software approach.  
-eg   
  
Reduced Lead Times: The reusing of an old nofiy user from a past system, will deacrease time needed to develop – know works??? Maybe go for something like feedback as good for iterations/rollouts  
  
Increased ROI: Resing an old or buying a component such as Receive Tips, will reduce development time. Ie it is cheaper to buy something than make it yourself in the long run.  
  
1.3  
  
Control  
How is control managed? The provide feedback/Provide Rating cannot occur if the alret user has not ran.  
How is control Transferred? Once the alert user runs, it transfers control to the provide feedback and provide Rating  
  
Hierarchy: Alert system – then sinpect work then request drone.  
  
Data  
Data communication:  
Data flow: Inspect work to saxx  
Active or passive: Active

Q2  
Data  
Data vs info: what is data, what is info  
  
Flow continuity  
Look at the data going in, processed, output.  
-What can be improved. What needed to evolve  
  
Direct vs Indirect, and how users need good info not table  
  
Consideration:  
This should be display cause

Architecture  
Define the core level view: Also defines the class relationships.  
  
Defines the major structural elements and their relationships.  
  
Define the patterns that can be used to achieve the requeiremtns for the application domain.  
  
Consideration  
Style for passive/active data

Interface:  
  
Defines how the software will look and feel (interaction UI).  
  
User InterfaceL: how user interacts with system  
Interior: How components interact  
Exterior: how components and DB interact

Consideration  
How should look

Component:  
  
Decide on how many components needed  
What their relationships will be  
What details they will conatain – input output – processes.  
  
Transforms the architectural structural elements into procedural description of software components.  
  
Component diagrams are made form class and behabioral diagrams.

**Data**

Data vs Indormation: Data is raw facts, info is processed.  
  
Flow Consistency  
We look at how data flwos in the system. What is the input daya, how is it used in system/process/manipulated, and what is outputted. WE look at how we can improve this process (more reliable. WE look at what data is needed to evolve system.

WE must also look at what type of data or information is the user seeing. Direct or indirect data. Ie tables bad bad.

**Consideration**

**Table bad – do this with data**

**Architecture**

The core view of the software: Define the relationships the classes have.   
  
Define the major sturcutral elements and their relationships **of the software**.

Defining what patterns to use that meet the requirements for the **specific domain.**

**Consideration**

Talk about data active/passive

**Interface**

The overall appearance and UI of the system (ie where button goes x)  
  
User Interface: User inreation with system  
Internal Interface: Interaction within system/components  
External Interface: DB interaction

**Consideration**

**Component**

Define what compoenets are needed  
Define what details the components contain  
Define the realtionshops between components.  
Define the components inputs/outputs.  
  
Information form the class and behaviorual diagrams are used to help create the componenet diagrams.

Transforms the structural elements of the architecture into procedural **description of software elements.**

**Consideration**

Q3.1  
Separation of concerns  
  
  
Abstraction  
-Makes details hidden. Don’t need to know everything to use it   
Procedural: sequential flow of operations of a function  
Data: Collection of data descriving an object

**Q3.2**

Data flow architecture:  
  
Can go directly from design to implementation phase: savee time = more money faster start.  
  
Transforms data from x->t>x we see this in diagram  
  
Needs active data.  
  
---------------------------------------------  
Data-Centerd architecture  
  
-Intergrating components easy  
-Passive data  
-when need DB to perform operations

Q3.3  
-Seqential process of data.  
  
Components do not know the implementation of nheiboring components.  
Components have an expected input – receive from X  
Components have an output – give to component T

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
DB  
Each component directly works with DB.  
Data is passive, ie component a doesn’t know theres been a change in data unless stored in DB.  
Very good for interopability