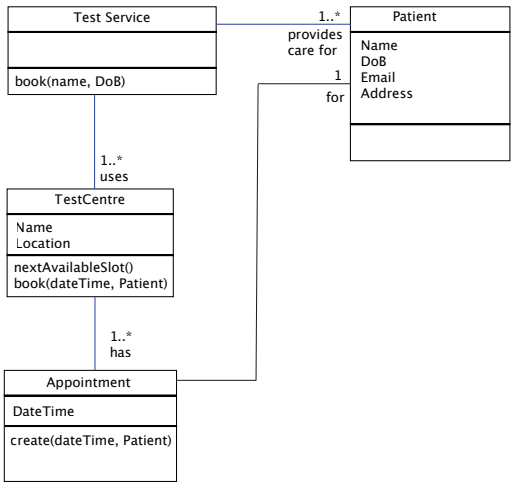
4M21 Software Engineering and Design Solutions: 2021/2022 Solutions

Elena Punskaya May 2022

Q1. **(a)** A class represents a key concept within the system. It encapsulates data and behaviour. Classes provide abstraction. A class can be used to create multiple instances, i.e. objects that can contain data and behave according to class definition. Given the same data (state) two independent instances of the same class will behave exactly the same. In production, a large number of objects are created, interact with each other, or are destroyed if no longer needed. [5%]

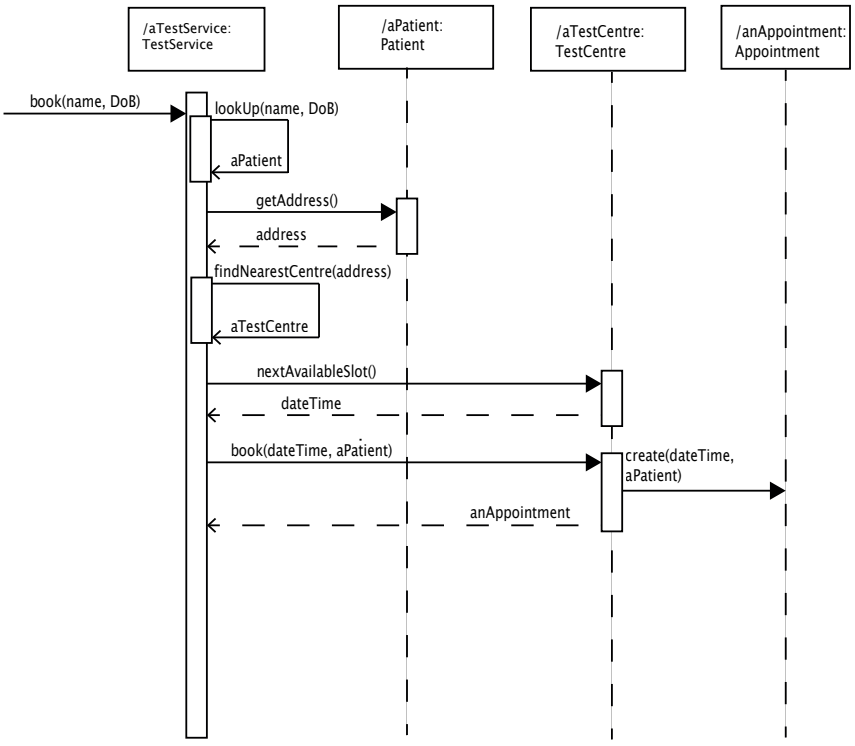
(b) One of possible solutions is presented below.

(i) Class diagram



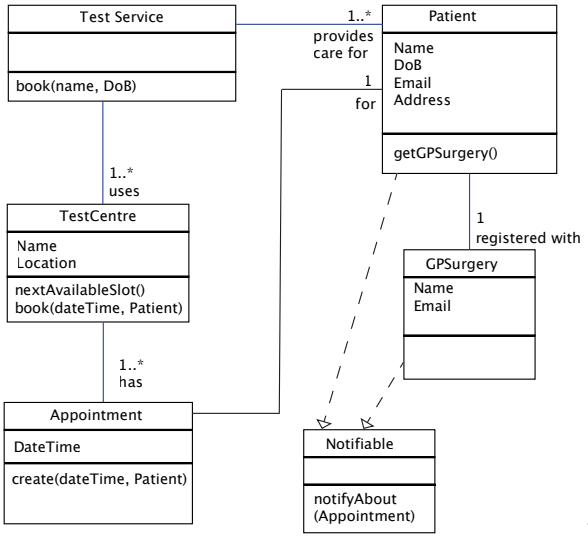
[35%]

(ii) Sequence diagram

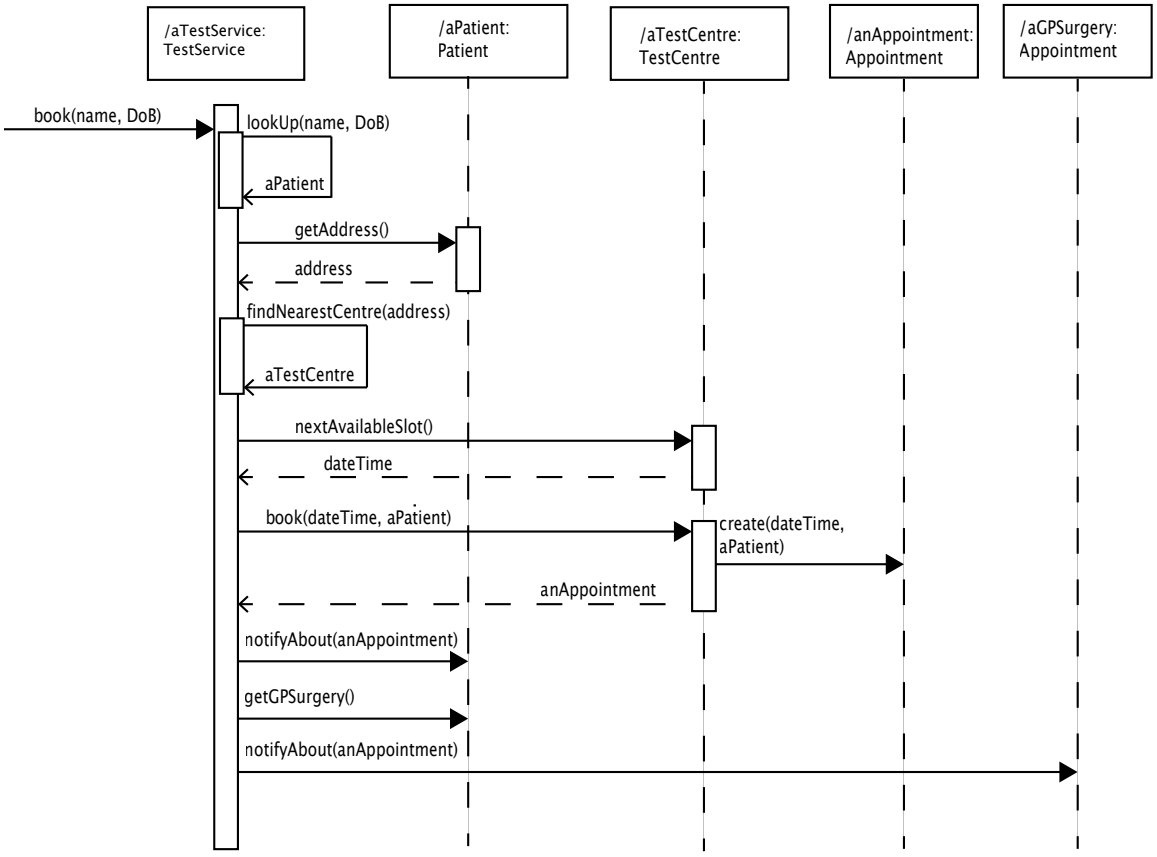


[30%]

(iii) Class diagram



Sequence diagram



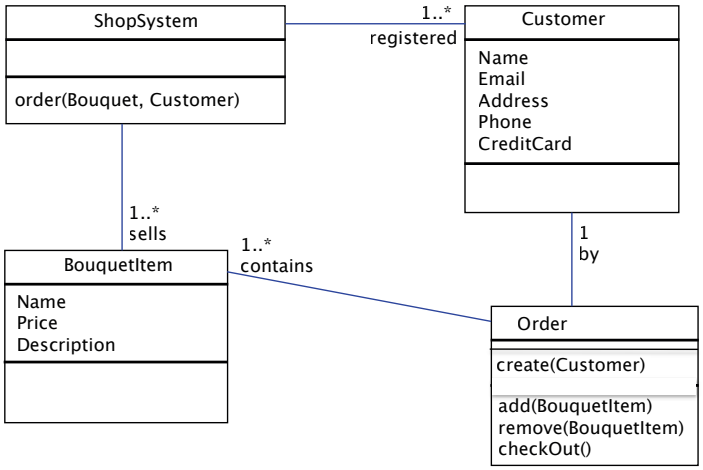
[30%]

Assessor's comments: The question was designed to test the ability to interpret the requirements independently, apply the main object-oriented design concepts in practice (in particular, principles of decoupling and abstraction) and the ability to communicate the design through class and sequence diagrams. It was the most popular question, and most students were able to go through an independent design process successfully, identifying the key concepts and communicating the outcome clearly using the standard notation. Not everyone was careful when working with both class and sequence diagrams as often sequence diagram did not correspond to the class diagram. Not everyone read the question carefully, and, as a result, some functionality was omitted / alternative functionality was provided.

Q2. (a) Although structurally somewhat similar, the intent is different. Composite allows to treat a composite (group of) leaves in the same way as a leaf. Decorator gives additional feature to a leaf and allows to treat "decorated" leaf in the same way as any other leaf. [10%]

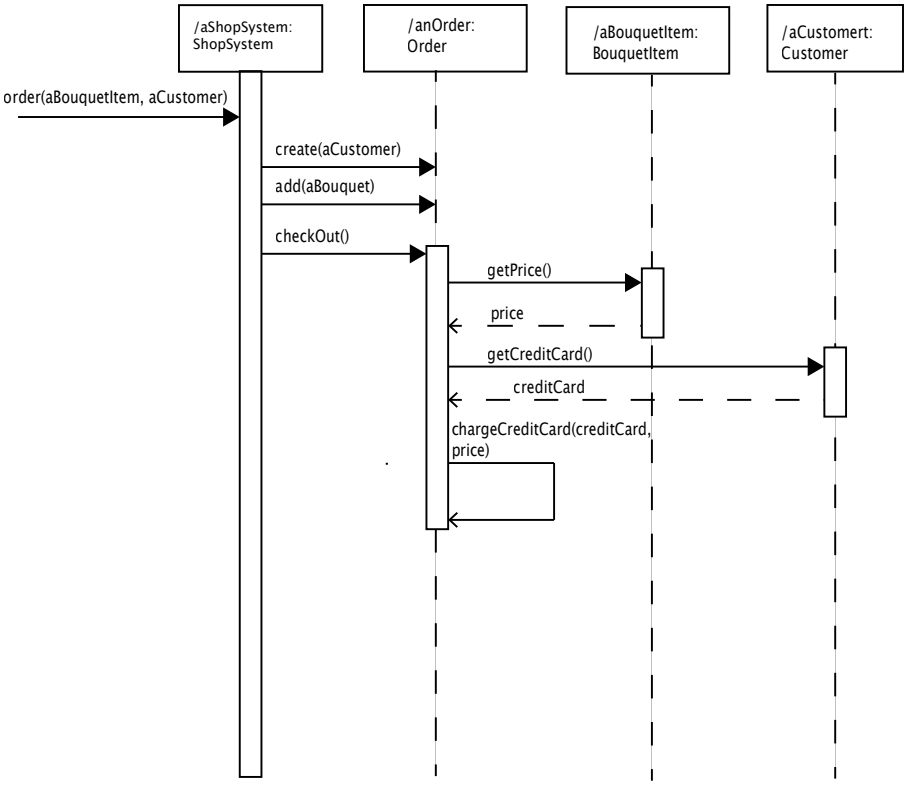
(b) One of possible solutions is presented below.

(i) Class Diagram



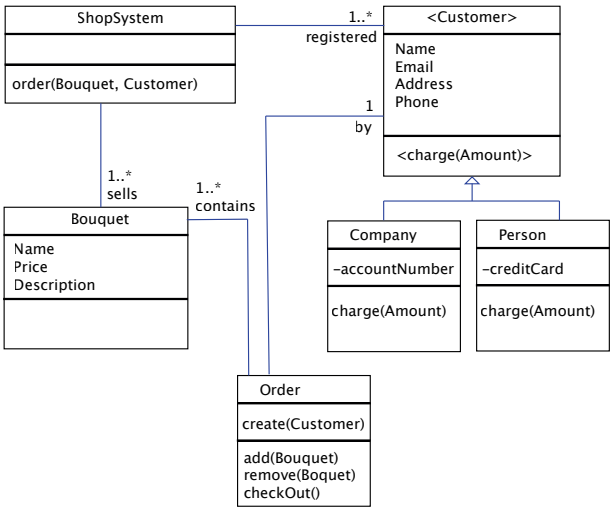
[20%]

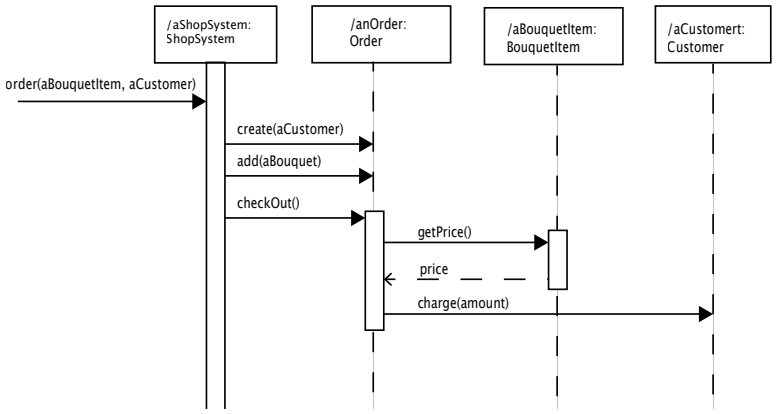
(ii) Sequence Diagram



[20%]

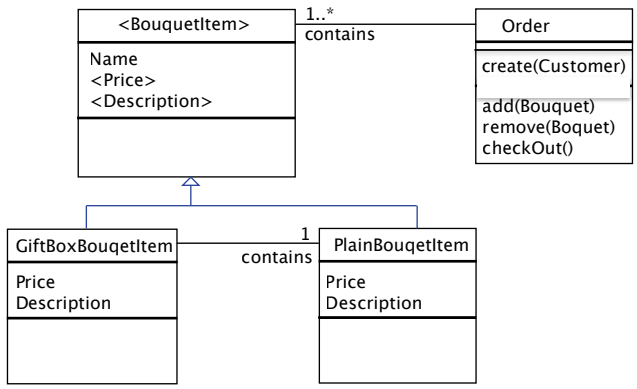
(iii) Class Diagram





[20%]

(iv) Class Diagram. Traditional implementation of Decorator would leave to "box in box in a box" game so is not applicable. At the very least it would have to be modified as follows.



[30%]

Assessor's comments: The question was on understanding of the key concepts of the object-oriented design such as polymorphism and inheritance and extending the design by identifying a common design problem and figuring out a reliable solution to it in a form of an appropriate design pattern as well as a correct way to apply it. Although less popular, the question was completed reasonably well, and the vast majority of students demonstrated their understanding of inheritance and were successful in identifying decorator pattern as the most appropriate one. However, not everyone was able to apply the decorator correctly in these specific circumstances, there were some inconsistencies between class and sequence diagrams, inaccuracies in UML notation and some candidates struggled with the concept of polymorphism.

Q3 (a) One of the possible answers presented below:

User-centred design including providing a good conceptual model, simplicity and predictability, designing for error and being understandable.

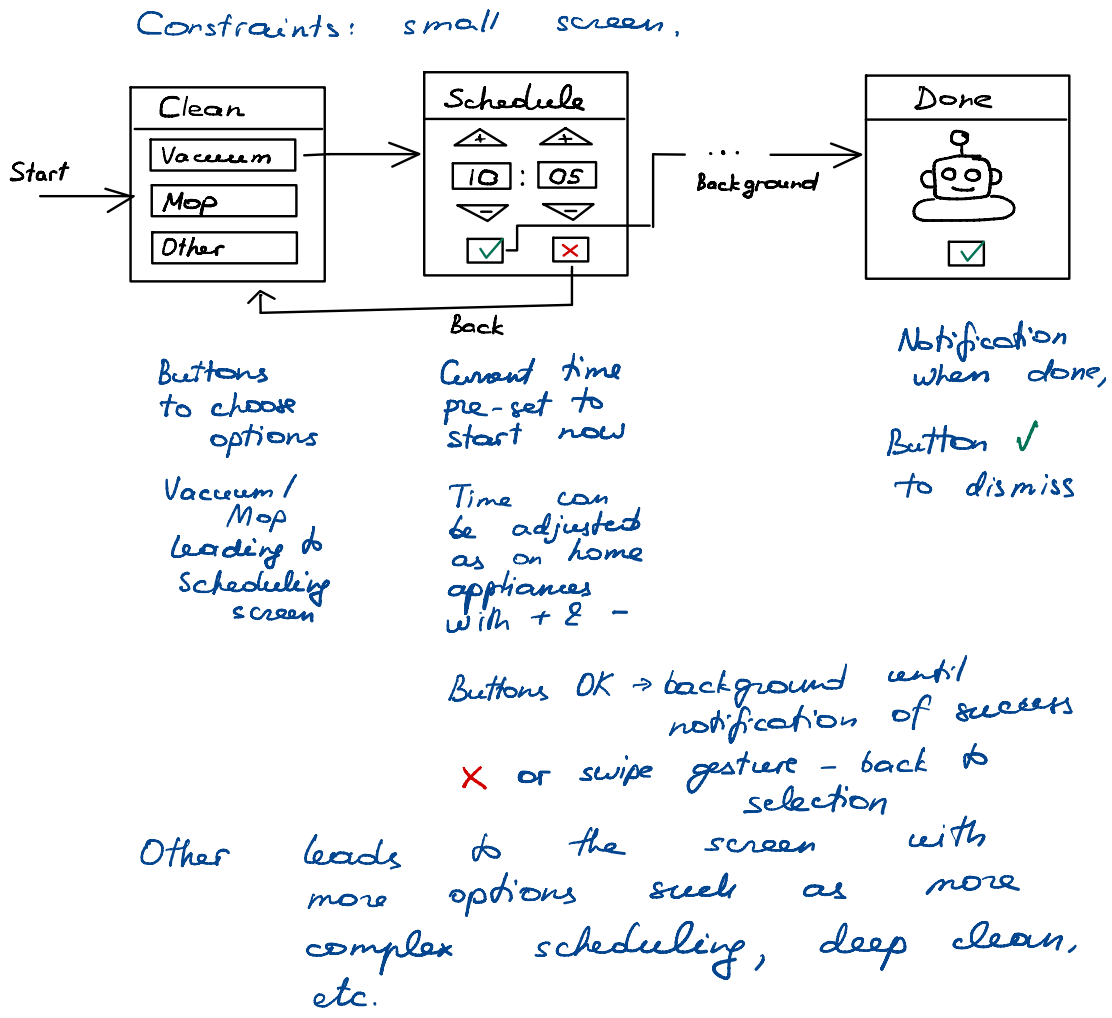
Activity-Centred Design, careful consideration of use cases.

Taking into account emotions and designing experiences.

[10%]

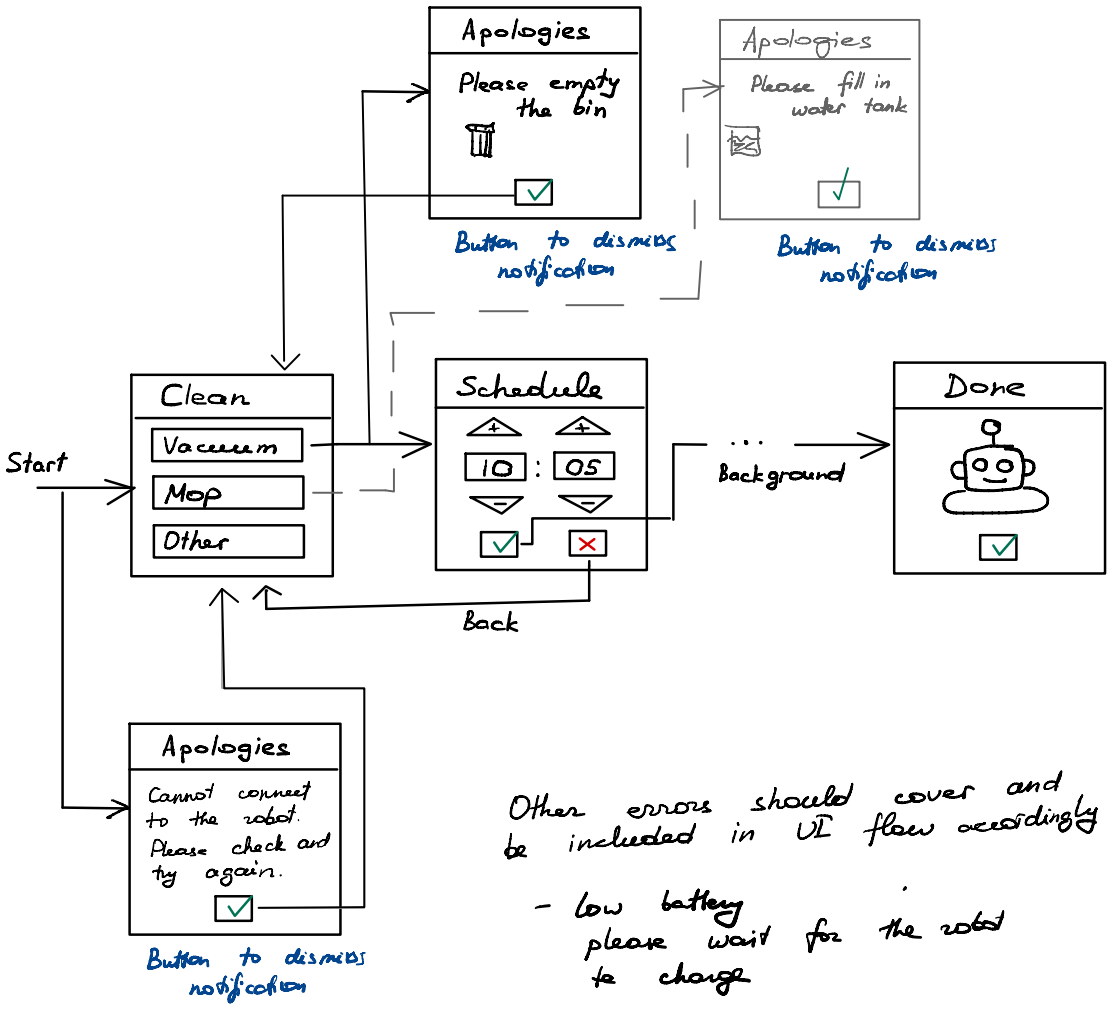
(b) One of possible solutions is presented below

(i)



[20%]

(ii)



Other errors should cover and be included in UI flow accordingly

- low battery
  please wait for the robot to charge

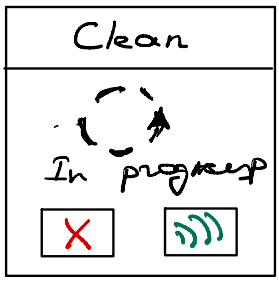
- cannot reach charging station (stuck / cannot move)
  please find the robot first

[20%]

(iii)

Intentions:

- check that all is well and the job is progressing as planned → confirmation screen
- stop cleaning
- find the robot



← Play sound to find robot

↑ Stop cleaning

Also handle anything that can go wrong such as

- full bin (if vacuuming)
- empty water tank (if mopping only)
- stuck and cannot continue
- delayed as extra recharge needed

[30%]

(iv)

Notifications

- full bin, please empty - can't proceed (additionally and optionally notify after the job is completed so that ready for the next one that can be scheduled from anywhere;
optional, when the user is on the same wifi, i.e. at home if possible and not at night time)
- robot is stuck
notify asap as can't proceed
  - repeat when the user is at home
- empty water tank
- delay due to extra charge

[20%]

Assessor's comments: This was a user interface design question that was answered well by most candidates. A popular question that candidates were able to complete without any major challenges, successfully optimising their design for a small smartwatch screen, although not everyone paid attention to the UI flow. Some candidates struggled with the concept of use cases and activity-based planning in part (b.iii) or ignored the pre-condition "after … [the users] instructed the robot to start the clean". Some candidates did not read the last part of the question carefully and suggested slight variations on the "successfully completed job" notifications even though the question stated that the user "found the notifications of successfully completed jobs unhelpful".

Q4. (a) Agile methodologies that could be given as an example: Extreme Programming, Test Driven Development, Feature Driven Development, Agile Modelling, Agile Unified Process, Dynamic System Development, Scrum among others.

The following could be noted… Agile techniques allow to incorporate a more iterative process to software development and therefore allow to adapt and grow software and incorporate changes. They do not necessarily allow to clarify system goals early in the process, and it might be difficult to charge customers for changes. If all requirements are not defined well in advance testing and delivery according to specification might be not suitable for critical products. Some of the more traditional project management tools might not work well with it although agile management tools are available. [15%]

(b) (i) Due to the nature of the application the core functions of the system are critical and have to work reliably from the first moment it goes live, the following considerations are important among others

- safety of people is involved, risk of multiple deaths, full risks analysis necessary

- expensive product, extremely high costs of potential recall

- regulatory requirements, testing to meet standards, requiring formal documentation

A formal waterfall model is recommended to develop the core functionality of the system. [20%]

(ii) Waterfall model follow a sequential process flowing steadily down the phases:

Requirements and Analyses: the environment and processes in which software will be used need to be analysed to establish operational parameters and required interfaces. Product requirements documentation is produced in compliance with relevant regulations.

Design: software architecture is defined to meet functional specifications. The design is documented in relation to requirements being addressed.

Implementation: code is produced and tested to perform functions according to specifications.

Testing: the test specification is developed and the product is tested Test results are captured.

Operations: requirements and guidelines are established for installation, updates and maintenance of the product

This project will typically require a Quality Management System in place and will be utilising Gantt chart for project management and Source Code Repository for code development. Other tools can be mentioned. [20%]

(iii)

Method: First, user interactions must be defined that are going to be tested. For example, the warning might be displayed to user when automated braking is activated or the sound might be activated, among other possibilities. Given the nature of the application a simulator should be used and the user can be explained how to use the simulator verbally and offered help if required. The users can be observed by the specialists, the session should be video recorded and feedback can then be captured via qualitative interviews by specialists. The session time could be around 20 mins.

User group: target user group can then be defined and segmented by age, gender, level of experience (driving), 5 user per group.

Scenarios: the user is asked to “drive” using simulator and “experience” 5 different types of automated braking system activated using simulator.

Report: includes conclusions and recommendations based on observations by a specialist, qualitative user feedback and charts/tables with any ranking/scoring users are asked to do. [20%]

(iv) The life cycle of the car is roughly two decades.

Would need to maintain multiple versions of the software going back over the years. This is costly. Fixes or new functionality on older models might be a challenge. Should try to limit the number of version and make sure fixes can be back-ported easily.

Security and safety are major concerns. Security and safety updates and patches are critical and will be a regular occurrence.

Safety standards tend to change. Safety fixes involving new functionality might be a challenge on older hardware. Might need backup solutions such as manual override.

Costs of maintenance of the software is very significant

[25%]

Assessor's comments: A reasonably straightforward question on software engineering methodologies and their application. Those who did attempt the question answered most parts well. Some candidates did not provide direct answers to the questions and instead made generic statements. Many candidates found part (b.iii) challenging and mixed up the concept of performance testing and usability studies, although majority of candidates were clearly familiar with the general methodology. Most candidates successfully identified critical nature of the application and as a result suggested a formal waterfall software development model, yet backtracked in the last part of the question and concentrated on agile even though the critical aspect of the applications did not change.