Chapter 2

2.1 Suggest the most appropriate generic software process model that might be used as a basis for managing the development of the following systems:

- A system to control antilock braking in a car
- A virtual reality system to support software maintenance
- A university accounting system that replaces an existing system
- An interactive travel planning system that helps users plan journeys with the lowest environmental impact

Generic software process model for:

- A system to control anti-lock braking in a car
  It is used for safety critical. Here the method is based on formal transformations and it would be the most appropriate. Partial credit was given for “incremental” or “spiral” if accompanied by a reasonable rationale. It certainly needs a plan-driven approach to development with the requirements carefully analysed. A waterfall model is therefore the most appropriate approach to use, perhaps with formal transformations between the different development stages.

- A virtual reality system to support software maintenance
  A virtual reality system would be cutting edge, and its usability would depend heavily on the quality of its user interface. Design/implementation that follows a “waterfall” or “formal” process model would be difficult here. An Evolutionary model seems ideal here, but full credit was given to “incremental” or “Spiral” with some UI prototyping. An agile process may be used.

- A university accounting system that replaces an existing system
  This system’s requirements are fairly well-known and will be used in an environment in conjunction with lots of other systems such as a research grant management system. A reuse-based approach is likely to be appropriate

- An interactive travel planning system that helps users plan journeys with the lowest environmental impact

System with a complex user interface but which must be stable and reliable. An incremental development approach is the most appropriate as the system requirements will change as real user experience with the system is gained.

2.2 Incremental software development could be very effectively used for customers who did not have a clear idea about the systems needed for their operations. Discuss

Incremental development is based on the idea of developing an initial implementation, getting feedback from users and others, and evolving the software through several versions until the required system has been developed. The approach can be either plan-driven, agile, or a mixture of these approaches. In Plan-driven, the system increments are identified in advance. In agile the early increments are identified but the development of later increments depends on progress and customer priorities.
Each increment of version of the system incorporates some of the functionality that is needed by the customer. The early increments of the system include the most important of most urgently required functionality. This means the customer can evaluate the system as a relatively early stage in the development to see if it delivers what is required. If not, then only the current increment has to be changed and, possible, new functionality defined for later increments.

2.3 Consider the integration and configuration process model shown in Figure 2.3. Explain why it is essential to repeat the requirements engineering activity in the process.

At first the requirements specification involve the initial requirements for the system that are proposed. These do not have to be elaborated in detail but should include brief descriptions of essential requirements and desirable system features.
During the requirement refinement stage, the requirements are refined using information about the reusable components and applications that have been discovered. The requirements are modified to reflect the available components, and the system specification is re-defined. Where modifications are impossible, the component analysis activity may be re-entered to search for alternative solutions.

2.4 Suggest why it is important to make a distinction between developing the user requirements and developing system requirements in the requirements engineering process

There is a fundamental difference between the user and the system requirements that mean they should be considered separately:

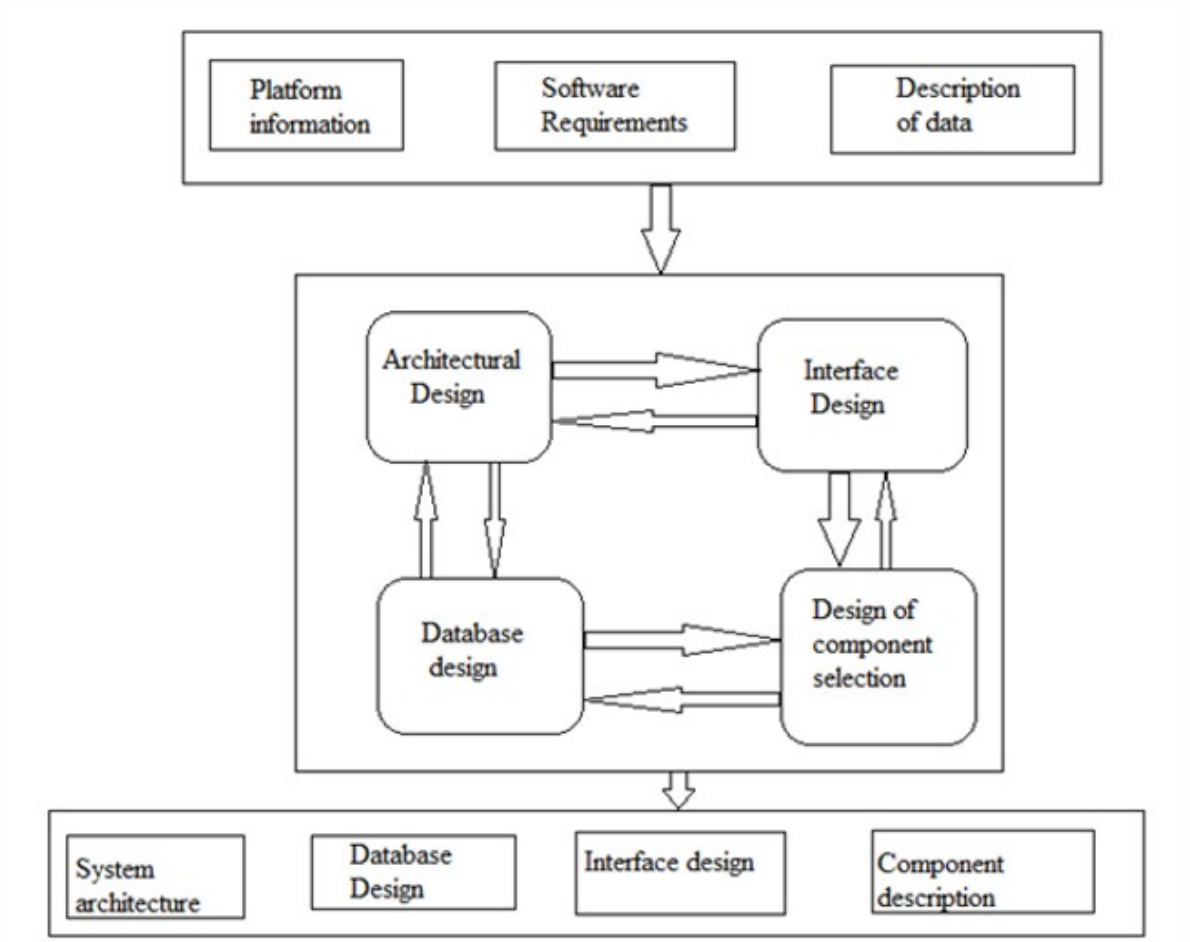
1. The user requirements are intended to describe the system's functions and features from a user perspective and it is essential that users understand these requirements. User requirements are abstract statements. Its described the systems functions and features of customer needs.
   System requirements are provided at a more detailed explanation of the procedure

2. They should be expressed in natural language and may not be expressed in great detail, to allow some implementation flexibility.
   System requirements are much more detailed and are intended to be a precise specification of the system that may be part of a system contract.

3. In user requirements, the starting level of the project, development team may put interest on gathering good information, identifying the important things and understanding them correctly.
   System requirements are implemented after user requirements have been established. The development of system requirements are important from the beginning of a project to end and beyond programmers must have the knowledge and skill to write program applications.

2.5 Using an example, explain why the design activities of architectural design, database design, interface design and component design are interdependent

Interdependent design activities:
Design of any software is a very important activity. It includes design of database, design of various components, user interface design and design of architecture. The diagram show an example to show interdependency of all design activities:



Consider an example of library management system and know how these design activities are interdependent:

1. Design of architecture: In this designing phase, the system's overall structure is defined. It also defines the distribution and relationship between principle components of the system.
    a. Design architecture of Library management describes the overall structure of library. It includes students who have access to the library, library administrative department and books linked to various departments etc.
    b. Therefore to issue a book from the library, these components must be communicated with each other.

2. Design of database: The structure of database with their respective representation is defined in this phase. Here, the user has to define that he is going to develop a new database or using the existing database

a. This phase is very important in Library management system because, huge amount of books related to different departments are stored in the database
b. Students are allowed to borrow the book with their credentials and needs to submit it as per the rules of the library administrator
c. The management must be consistent when books are borrowed and returned from the database
d. If the database is well designed, then it is easy to search and maintain the book information

3. Design of interface: The interface among different components of the system is defined here. It is possible that one component is again used for any implementation in the system.
    a. This phase defines various possible interfaces to allow and access the books to students
    b. Every student has library-ID or roll number to access the books. With these credentials, the student can directly go to the library and borrow the book or they may use the library web portal
    c. With the use of a library-ID or roll number, the student can search the availability of required books, and can see the due time for submission
    d. Therefore, interface is an important phase while designing any software

4. Component design and their selection: The reusable components which match the requirements of the new system are defined here:
    a. If a student requires a book which is not available in the library, the administrator should check for the matching books to satisfy the student request.
    b. If no match is found, management must purchase a new book which is requested from the student

    Therefore, to effectively manage the library and fulfil the student requirements, library management should follow all the design activities mentioned above.
    Hence, it can be said that all these activities are interdependent to each other.

2.6 Explain why software testing should always be incremental, staged activity. Are programmers the best people to test the programs that they have developed?

Software testing
Testing is defined as the process in which each program is tested for making sure that each program is functioning correctly. It is an incremental theatrical activity. The points to show the reasons are:

1. The complete program is divided into small modules and each module is checked undependably. Then, all programs are collectively tested for unit testing.
2. After those programs are tested in group and after that the whole system is tested. CASE tools are used for testing the occurrence of syntax errors which are generally grammatical errors. Desk checking is used for testing the occurrence of logical errors which makes a program to produce incorrect results.
3. A Beta version of the application is released and an end user checks it.

Thus, it can be said that it is an incremental theatrical activity.

NO, the programmers are not the best persons to test any program:

1. A programmer has skill to develop the program but they don't have the best skills to test the developed program
2. It is very difficult to find errors and mistakes in own developed program. Thus, testing should be conducted by testers and end users, so that all possible errors can be found.

2.7 Imagine that a government wants a software program that helps to keep track of the utilization of the country's vast mineral resources. Although the requirements put forward by the government were not very clear, a software company was tasked with the development of a prototype. The government found the prototype impressive, and asked it be extended to be the actual system that would be used. Discuss the pros and cons of taking this approach

The pros of this approach are:

1.It allows changes to be made to the prototype

2.The customer can be shown the protoype and modifications can be made before the design is created

3.It leads to better implementation

4.The customer is satisfied

The cons of this approach are:

1.Additional costs must be invested to create the prototype

2.The product might get delayed as creating prototypes and modifying them takes a lot of time.

2.8 You have developed a prototype of a software system and your manager is very impressed by it. She proposes that it should be put into use as a production system, with new features added as required. This avoids the expense of system development and makes the systems immediately useful. Write a short report for your manager explaining why prototype systems should not normally be used as production systems

A software prototype can be used in a software development process to help anticipate changes that may be required:

1. In the requirements engineering process, a prototype can help with the elicitation and validation of system requirements
2. In the system design process, a prototype can be used to explore software solutions and in the development of a user interface for the system

The objective of prototyping should be to develop the user interface, to develop a system to validate functional system requirements or to develop a system to demonstrate the application to managers. The same prototype cannot meet all objectives.

To reduce prototyping costs and accelerate the delivery schedule, some functionality is left out of the prototype. Non-functional requirements such as response time and memory utilisation may have been relaxed. Error handling and management may have been ignored and standards of reliability and program quality may have been reduced.

Provision should be made for user training. Potential users need time to become comfortable with the new system and settle into a normal pattern of usage. Only then will requirement errors and omissions be found.

A prototype is often used as complete systems due to the pressures of development but they have several faults:

1. The user interface may be minimal and not intuitive
2. There may be no error detecting or handling code
3. Such error messages may be vague
4. Generally, prototypes are not viewed as high quality products, but just tools to aid the development process.

2.9 Suggest two advantages and two disadvantages of the approach to process assessment and improvement that is embodied in the SEI's Capability Maturity framework

Advantages:

1. It focused attention on the software engineering processes and practices that were used
2. Led to significant improvements in software engineering capabilities

Disadvantages:

1. There is too much overhead in formal process improvement for small companies
2. Maturity estimation with agile processes is difficult

2.10 Historically, the introduction of technology has caused profound changes in the labour market and, temporarily at least, displaced people from jobs. Discuss whether the introduction of extensive process automation is likely to have the same consequences for software engineers. IF you don't think it will, explain why not. If you think that it will reduce job opportunities, is it ethical for the engineers affected to passively or actively resist the introduction of this technology?

Due to the introduction of extensive process automation, they have the potential to reduce the human error in creation of code and made it meet the precise

syntax and other constrains. It also has the potential to produce similar or better software than that produced conventionally by relatively scarce skills software development talent and of course will reduce the cost.

Automation will lead to a huge use of standardized components, and thus increasing software reliability and lessen the cost for future software maintenance. Software engineers will be recognized because of the production of the less- interesting, reduce more mechanical task software that engineers have to perform and thus can lead them to be more creative in the task they was assigned. Not to be forgotten, automation also assists software to address the primary issues in the software development process such as complexity, reliability and productivity. This effect is different when mention about labor market. Since the automation will lessen the need of human sensory and mental requirements of work. In a wide range industries beyond manufacturing like telephone operators, the electrocardiography or radiography used in medical process and Automated teller machine have reduced the need of human intervention.

Chapter 3

3.1 At the end of their study program, students in a software engineering course are typically expected to complete a major project. Explain how the agile methodology may be very useful for the students to use in this case

The agile methodology is team centered so it would help them interact with each other. It also encourages releases in phases so that the team can show the professor updates as they work.
The processes of specification, design and implementation are interleaved. There is no detailed system specification and design documentation is minimised.

3.2 Explain how the principles underlying agile methods lead to the accelerated development and deployment of software.

The principles underlying agile development are:

1. Individual and interactions over processes and tools
   By taking advantages of individual skills and ability and by ensuring that the development team know what each other are doing, the overheads of formal communication and process assurance are avoided. This means that the team can focus on the development of working software

2. Working software over comprehensive documentation
   This contributes to accelerated development because time is not spent developing, checking and managing documentation. Rather the programmer's time is focussed on the development and testing of code.

3. Customer collaboration over contract negotiation.
   Rather than spending time developing, analysing and negotiating requirements to be included in a system contract, agile developers argue that it is more effective to get feedback from customer's directly during the development about what is required. This allows useful functionality to be developed and delivered earlier than would be possible if contracts were required.

4. Responding to change over following a plan.
   Agile developers argue that being responsive to change is more effective than following a plan-based process because change is inevitable whatever process is used. There is significant overhead in changing plans to accommodate change and the inflexibility of a plan means that work bay be done that is later discarded.

3.3 Extreme programming expresses user requirements as stories, with each story written on a card. Discuss the advantages and disadvantages of this approach to requirements description

Advantages of stories:

1. They represent real situations that commonly arise so that systems will support the most common user operations
2. It is easy for users to understand and critique the stories

3. They represent increments of functionality – implementing a story delivers some value to the user

Disadvantages of stories:

1. They are liable to be incomplete and their informal nature makes this incompleteness difficult to detect
2. They focus on functional requirements rather than non-functional requirements
3. Representing cross-cutting system requirements such as performance and reliability is impossible when stories are used.
4. The relationship between the system architecture and the user stories is unclear so architectural design is difficult

3.4 In test-first development, tests are written before the code. Explain how the test suite may compromise the quality of the software system being developed.

The test suite could become outdated by the time the actual code is written and not test every vulnerability that the code may have. If the testes are not reviewed and further tests written after development, then undetected bugs may be delivered in the system release.

3.5 Suggest four reasons why the productivity rate of programmers working as a pair might be more than half that of two programmers working individually

1. Pair programming leads to continuous informal reviewing. This discovers bugs more quickly than individual testing
2. Informal sharing in pair programming is implicit – it happens during the process. This reduces the need for documentation and the time required if one programmer has to pick up another's work. Individual programmers have to spend time explicitly sharing information ant they are not being productive when doing so.
3. Pair programming encourage refactoring (the code must be understandable to another person). This reduces the costs of subsequent development and change and means that future changes can be made more quickly. Hence efficiency is increased

4. In pair programming, people are likely to spend less time in fine-grain optimisation as this does not benefit the other programmer. This means that the pair focus on the essential features of the system which they can then produce more quickly

3.6 Compare and contrast the Scrum approach to project management with conventional plan –based approaches. Your comparison should be based on the effectiveness of each approach for planning the allocation of people to projects, estimating the cost of projects, maintaining team cohesion, and managing changes in project team membership

- Planning allocation of people to projects
  - Scrum: Scrum handles people allocation informally. Team members "bid" for features from the product backlog to implement if they think that their expertise is appropriate. Alternatively, the tasks can be allocated by the Scrum master.
    There is no formal mechanism in Scrum for planning for project members with very specific expertise to be temporarily allocated to a team. This need must be identified by the Scrum master and he has to discuss how the expertise can be made available.
  - Plan-based development: A project plan is used to identify the parts of the system to be delivered and these are specified in the requirements document. The expertise required for each part can then be identified and the allocation of people to projects planned on that basis

- Estimating project costs
  - Scrum: Project costs are estimated based on the required delivery date for the software and people working in the Scrum team. The functionality of the system is adjusted sot that some working system will always be delivered for the original cost estimation. Of course, this may not be adequate for the customer and they have to become involved in rescheduling the delivery of the system
  - Plan-based: Project costs are based on an analysis of the functionality specified in the requirements document as well as the non-functional requirements of the system. They may be adjusted to reflect team size and delivery schedule. It is normal for costs to be

underestimated and the final project to cost much more than originally estimated. An average cost for team members is assumed.

- Maintaining team cohesion
  - Scrum: Team members meet daily face to face or electronically. Extensive formal discussion and communications are encouraged. Team members negotiate work to be done from the project backlog. This all leads to a share feeling of product ownership and a very cohesive team.
  - Plan-based: Team cohesion is the responsibility of the project manager and he has to take explicit actions to encourage this. The general approach relies on formal meetings that are relatively infrequent and this does not lead to the development of a cohesive team

- Managing changes in project team membership
  - Scrum: This is a topic that is rarely discussed in Scrum but is a fundamental problem because so much information is informal and reliant on people remembering what has been agreed. When someone leaves, it can be very difficult to bring a replacement team member up to speed, especially if very little project documentation is available.
  - Plan-based: The project management plan is based around expertise rather than individuals and project documents should be available. Therefore, if a team member leaves, then a new team member with comparable expertise can read what has been done and after understanding this, should be able to serve as a replacement.

3.7 To reduce costs and the environmental impact of commuting, your company decides to close a number of offices and to provide support for staff to work from home. However, the senior management who introduce the policy are unaware that software is developed using Scrum. Explain how you could use technology to support Scrum in a distributed environment to make this possible. What problems are you likely to encounter using this approach?

Use of technology:

- Videoconferencing between the product owner and the development team
- The scrum master should be located with the development team so that he is aware of everyday problems
- The product owner should visit the developers and try to establish a good relationship with them. It is essential that they trust each other
- Real-time communications between team members for informal communication particularly instant messaging and video calls
- Continuous integration, so that all team members can be aware of the state of the product at any time
- A common development environment for all teams

Problems we can encounter:

- Development requires daily meetings but in distributed environment this is not possible
- A Communication gap bay occur between team members
- Changes can lead to slow the entire development of the project
- The most important benefit of pair programming for detection and evaluation of errors will be lost.

3.8 Why is it necessary to introduce some methods and documentation from plan-based approaches when scaling agile methods to larger projects that are developed by distributed development teams?

1. *Project planning* is often essential when developing software with larger teams to (a) ensure that the right people are available when they are needed to be involved in the development process and
(b) ensure that the delivery schedules of different parts of the system developed by different teams are aligned. This means that if Part A depends on Part B, the schedule should ensure that Part B is developed before Part A.

2.Requirements analysis and documentation is important to decide how to distribute the work across teams and to ensure that each team has some understanding of what other teams are doing.

3.Design documentation especially interface specifications are important so that teams can develop independently without having access to software that is under development.

4. Risk management may be required to ensure that all of the teams understand the risks faced and can organize their work to minimize these risks. Risk management may also be useful to cope with different delivery schedules used by different teams.

3.9 Explain why agile methods may not work well in organizations that have teams with a wide range of skills and abilities and well-established processes.

Having to switch from their well-established process to a new process might set them back on time that they could use to develop.

3.10 One of the problems of having a user closely involved with a software development team is that they "go native." That is, they adopt the outlook of the development team and lose sight of the needs of their user colleagues. Suggest three ways how you might avoid this problem, and discuss the advantages and disadvantages of each approach.

1. ***Involve multiple users in the development team.*** Advantages are you get multiple perspectives on the problem, better coverage of user tasks and hence requirements and less likelihood of having an atypical user. Disadvantages are cost, difficulties of getting user engagement and possible user conflicts.

2. ***Change the user who is involved with the team*.** Advantages are, again, multiple perspectives. Disadvantages are each user takes time to be productive and possible conflicting requirements from different users.

3. ***Validate user suggestions with other user representatives*.** Advantages are independent check on suggestions; disadvantage is that this slows down the development process as it takes time to do the checks.

Chapter 4

4.1 Identify and briefly describe the four types of requirements that may be defined for a computer-based system

1. User Requirements: which is the requirements that are statements in natural language plus diagrams of the services the system provides and its operational constraints.
2. System Requirements: A structured document setting out detailed description of the systems functions, services and operational constraints. Define what should be implemented. It may be part of a contract between client and contractor.
3. Functional Requirements: these are the statement of the services the system should provide, how the system should react to particular input and how the system should behave in particular situations.
4. Non-functional requirements: Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards etc often these are applied to the system as a whole rather than individual features or services.

4.2 Discover ambiguities or omissions in the following statement of the requirements for part of a drone system intended for search and recovery:
The drone, a quad chopper, will be very useful in search and recovery operations, especially in remote areas or in extreme weather conditions. It will click high-resolution images. It will fly according to a path preset by a ground operator, but will be able to avoid obstacles on its own, returning to its original path whenever possible. The drone will also be able to identify various objects and match them to the target it is looking for

- How will the drone determine when to take a picture
- Where will the pictures be stored
- What is the memory capabilities of the storage facility
- Will the drone be able to take pictures at night
- Can the ground operator change the path during flight

- What will happen if the drone cannot locate the original path after avoidance of obstacles
- Can more than one target be uploaded
- What will happen when the target is found
- Is a location saved on where the target is found
- How will the drone notify the ground operator that the target is found
- Can the ground operator request the drone to return before the preset path is complete

4.3 Rewrite the above description using the structured approach described in this chapter. Resolve the identified ambiguities in a sensible way

Function	Search and Recovery Drone
Description	Drone used during search and recovery to find the target on a set path
Input	Target image and set path
Source	Ground operator
Outputs	High-resolution images matched to target
Destination	
Action	The drone fly according to the preset path and take high-resolution images until the target is matched to the images taken
Requires	Preset path and target image
Precondition	Target image must match images taken
Post condition	Target is found
Side effects	Target is not found or matched

User Requirements:
A drone to be used during search and recovery operations. The drone should fly according to a preset path and click high-resolution images and match them to the target it is looking for.

System Requirements:
Functional Requirements:
The drone should fly according to a preset path set by a ground operator

The drone should be able to avoid any obstacle in its path and find the original path again
The drone should take high-resolution pictures every 5-10 seconds
When the drone identify the target it should send the GPS location to the ground operator
Between 1 and 10 targets should be loaded to the memory of the drone
The pictures clicked by the drone is immediately downloaded to the ground operator and not saved on the drone's memory in case the drone is lost
The ground operator should be able to track the drone's GPS coordinates at all times.
A new alternative path to be advised during flight by the ground operator
When the target is found and the ground operator, the ground operator should be able to instruct the drone to abort the current path and return back to base

4.4 Write a set of Non-functional requirements for the drone system, setting out its expected safety and response time

The range that the drone can travel away from base to still pick up signal. When range is exceeded a warning signal sent to ground operator. Path should be changed to get drone back into signal range.
Response time to path changes within 10 – 30 seconds
Night images
Infra-red images to pick up heat during night
Feedback to the ground operator should be within 10-30 seconds

4.5 Using the technique suggested here, where natural language descriptions are presented in a standard format, write plausible user requirements for the following functions.

1. An unattended petrol pump system that includes a credit card reader. The customer swipes the card through the reader, then specifies the amount of fuel required. The fuel is delivered and the customer's account debited.

Function	Issue gas

Description	Issue gas by deducting the payment from the user's credit card based on the monetary amount or amount of litres advised by the user
Input	Monetary amount or litres with credit card details
Source	Credit card and input details
Outputs	The required gas is issued
Destination	
Action	Gas filling station is at zero state User swipe credit card and advise either the monetary amount or litres System validate the user's credit card and credit availability Validation process should be returned within 5 seconds When the card validation is returned as success, the system indicates to the user to fill the car When the advised monetary amount or litres are reached the system stops with the filling action and advise the user to return the nozzle in the original state The system debits the credit card and issue a receipt. The receipt should be issued within 5-10 seconds The user interface should be easy to navigate and understand The display should output real-time the amount of litres and amount already pumped. The display should be bright during night time hours The nozzle should be able to sense when the fuel tank is overflowing and stop pumping
Requires	The monetary amount or litres and a credit card
Precondition	Credit card is valid and credit is available
Post condition	The requested gas is issued to the user
Side effects	None

The cash-dispensing function in a bank ATM.

Function	Issue cash
Description	Issue cash by validating the user's debit card and Pin and debiting the account by the specific amount advised by the user

Input	Monetary amount with debit card details
Source	Debit card and advised amount
Outputs	The advised amount of cash is issued to the user
Destination	Debit account with amount
Action	Initially the cash dispensing system is at zero state User swipes/inserts card and enter the secret pin associated with the card System validates the pin entered against the card If validation is successful the system request the amount to be withdrawn User advise the amount The system validate the amount against the account balance If the balance is sufficient, the cash is issued to the user. If the balance is insufficient the system display the amount available for withdrawal and allow the user to change the amount requested. The user can cancel the transaction at any time before the cash is issued When cash is issued successfully the bank balance is debited with the amount issued. System interface should be intuitive The system should alert the user audibly when transactions are completed and prompt the user when the card is left in the reader
Requires	An amount and a valid card
Precondition	Valid debit card and pin
Post condition	The cash is issued to the user
Side effects	None

In an internet banking system, a facility that allows customers to transfer funds from one account held with the bank to another account with the same bank

Function	Internal fund Transfer
Description	Transfer funds by deducting the amount from the user's account and input it into the account specified by the user
Input	Account number of receiver
Source	Transfer fund from source account to destination account for a

	specific amount
Outputs	The required funds are transferred to the destination account
Destination	Amount input and destination account
Action	Initially the fund transfer system is as zero state User select the fund transfer menu option User specify the from account from the list of available accounts User specify the amount to be transferred User specify the account to where the funds should be transferred System validate the destination account details specified System validate the specified amount against the balance available in the from account When validation is successful the funds is debited against the from account and credited to the destination account Notification sent to the user that the transaction was successful Notification sent to the receiver to notify of the transfer User logs out of the system and system returns to zero state
Requires	Sign on onto the system Destination account details Amount
Precondition	User have valid online banking log in details The destination account details are correct Available funds in the from account
Post condition	The required funds are transferred from the source to the destination account
Side effects	None

4.6 Suggest how an engineer responsible for drawing up a system requirements specification might keep track of the relationships between functional and non-functional requirements

Keeping track of the relationship between functional and non-functional requirements is difficult because non-functional requirements are sometimes

system level requirements rather than requirements which are specific to a single function or group of functions.
One approach that can be used is to explicitly identify system-level non-functional requirements that are associated with a functional requirement and list them separately. All system requirements that are relevant for each functional requirement should be listed. They can be related by including them in a table:

Functional requirement	Related non-functional system requirement	Non-functional requirement
The system shall provide an operation which allows operators to open the release valve to vent steam into the atmosphere	Safety requirement: No release of steam shall be permitted if maintenance work is being carried out on any steam generation plan	Timing requirement: the valve must open completely within 2 seconds of the operator initiating the action

Another way the engineer can keep track of the relationship between requirements is through a requirements verification document. This document includes all requirements and has the procedures that will go into validating each one is correct. A requirements matrix within this document would show how the requirements relate to each other.

Doing functional and non-functional requirements at the same time should help prevent overlap and help the engineer connect new functional requirements to previous non-functional requirements.

4.7 Using your knowledge of how an ATM is used, develop a set of use cases that could serve as a basis for understanding the requirements for an ATM system

Use cases are generally a set of interaction between the User with the system to generate a desired output
A use case diagram is a graphical representation of all use cases that interacts with the system.

Withdraw cash:

Actors	Customer, ATM, accounting system
Inputs	Customer’s card, PIN, Bank account details

Outputs	Customer's card, Receipt, bank account details
Normal operation	Customer inputs his card into the machine He is prompted for a pin which is entered on the keypad If correct, he is presented with a menu of options The withdraw cash option is selected The customer is prompted with a request for the amount of cash required and enters the amount If there is sufficient funds in the account, the cash is dispensed, a receipt is printed and the account balance is updated. The card is returned to the customer who is prompted by the machine to take the card
Exception	Invalid card: Card is retained by machine. Customer is advised to seek advise Incorrect pin: customer is requested to rekey Pin. If incorrect after 3 attempts, card is retained by machine and customer is advised to seek advise Insufficient balance: Available balance is displayed to customer and allow customer to renter a valid amount

Display balance:

Actors	Customer, ATM, Accounting system
Inputs	Customer's card, PIN, Bank account details
Outputs	Customer's card, Receipt, bank account details
Normal operation	The customer authenticates using his card and associated PIN Customer select Display balance option The Current balance of their account is displayed on the screen Option is given to print the balance on a receipt Other menu options displayed to customer or terminate the transaction The card is returned to the customer
Exception	Invalid card: Card is retained by machine. Customer is advised to seek advise Incorrect pin: customer is requested to rekey Pin. If incorrect after 3 attempts, card is retained by machine and customer is advised to seek advise

Print statement:

Actors	Customer, ATM, Accounting system
Inputs	Customer's card, PIN, Bank account details
Outputs	Customer's card, Printed statement, bank account details
Normal operation	The customer authenticates using his card and associated PIN Customer select Print statement option The last five transaction on their account is printed Receipt with transaction is issued to the customer Other menu options displayed to customer or terminate the transaction The card is returned to the customer
Exception	Invalid card: Card is retained by machine. Customer is advised to seek advise Incorrect pin: customer is requested to rekey Pin. If incorrect after 3 attempts, card is retained by machine and customer is advised to seek advise

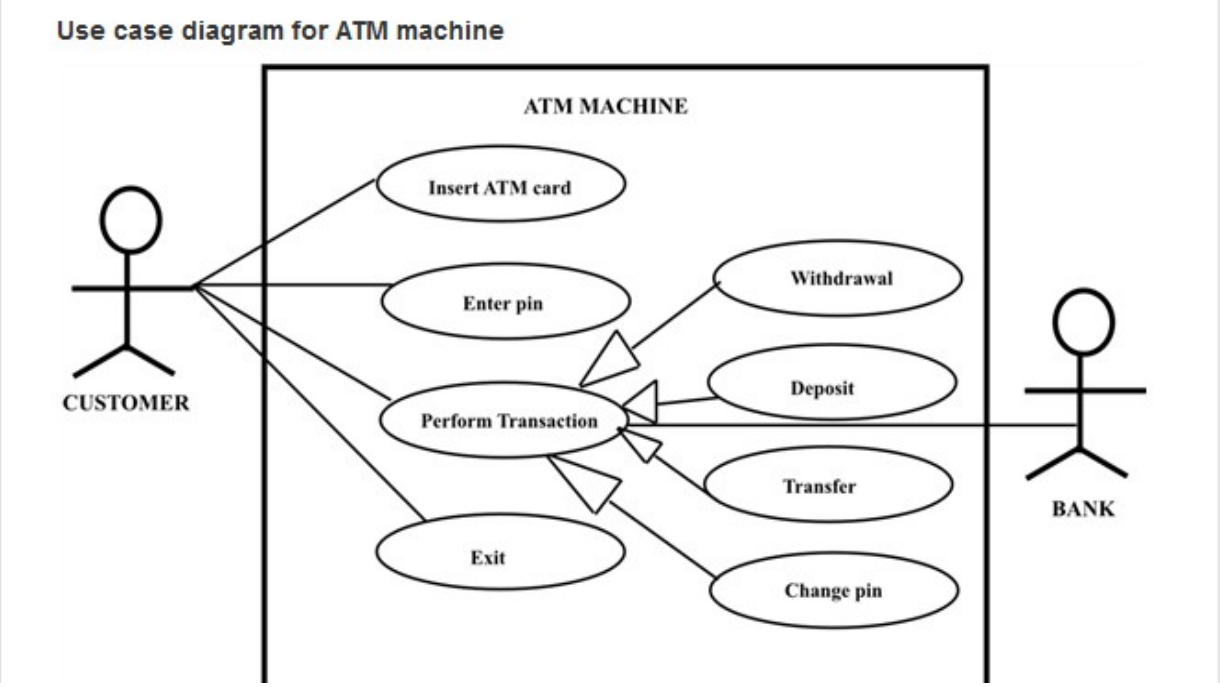
Change PIN

Actors	Customer, ATM, Accounting system
Inputs	Customer's card, PIN, Bank account details
Outputs	Customer's card, Printed statement, bank account details
Normal operation	The customer authenticates using his card and associated PIN Customer select Change PIN option He is prompted twice to input the new PIN System validate that the input are the same The new PIN number is encrypted and stored on the card Other menu options displayed to customer or terminate the transaction The card is returned to the customer
Exception	Invalid card: Card is retained by machine. Customer is advised

	to seek advise Incorrect pin: customer is requested to rekey Pin. If incorrect after 3 attempts, card is retained by machine and customer is advised to seek advise Pins do not match: The customer is prompted to repeat the process to reset the PIN

Deposit cash:

Actors	Customer, ATM, accounting system
Inputs	Customer's card, PIN, Bank account details
Outputs	Customer's card, Receipt, bank account details
Normal operation	Customer inputs his card into the machine He is prompted for a pin which is entered on the keypad If correct, he is presented with a menu of options The Cash Deposit option is selected The customer is prompted for the amount of cash to be deposited and inputs the amount. He is issued with a deposit envelope in which he should put the cash then return it back into the machine The account balance is updated with the amount deposited but marked as uncleared funds. Once the bank has verified the amount the funds are released as cleared funds. A receipt is issued The card is returned to the customer who is prompted by the machine to take the card
Exception	Invalid card: Card is retained by machine. Customer is advised to seek advise Incorrect pin: customer is requested to rekey Pin. If incorrect after 3 attempts, card is retained by machine and customer is advised to seek advise No cash deposited within 1 minute of envelope being issued, the transaction is terminated and card returned to the customer



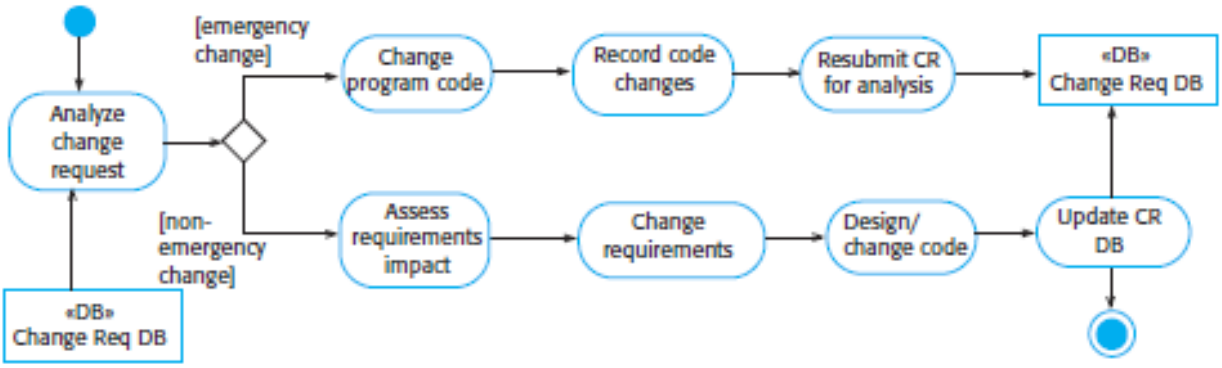
4.8 To minimise mistakes during a requirements review, an organisation decides to allocate two scribes to document the review session. Explain how this can be done

The scribe (or recorder) has to record each defect mentioned and any suggestions for process improvement. In practice it is often the author who plays this role, ensuring that the log is readable and understand-able

4.9 When emergency changes have to be made to systems, the system software may have to be modified before changes to the requirements have been approved. Suggest a model of a process for making these modifications that will ensure that the requirements document and the system implementation do not become inconsistent

The following diagram shows a change process that may be used to maintain consistency between the requirements document and the system. The process should assign a priority to changes so that emergency changes are made but these changes should then be given priority when it comes to making modifications to the system requirements. The changed code should be an input to the final

change process but it may be the case that a better way of making the change can be found when more time is available for analysis.



4.10 You have taken a job with a software user who has contacted your previous employer to develop a system for them. You discover that your company's interpretation of the requirements is different from the interpretation taken by your previous employer. Discuss what you should do in such a situation. You know that the costs to your current employer will increase if the ambiguities are not resolved. However, you also have a responsibility of confidentiality to your previous employer

In a situation where the interpretation of requirements by the system developer is different from that of the user, the company is to be suggested for requirements validation. After discovering that there is a difference in the interpretation of requirements, it is to be verified with the concerned authority whether the requirements validation has been performed or not. If not, it is to be suggested to implement requirement validation processes.

Requirement validation is the process of checking that requirements actually define the system what they want. It overlaps with analysis as it is concerned with finding problems with the requirements. During this phase they can discover that there is a difference in interpretation. Also it can be advised to conduct a formal meeting or informal meeting in which they can discuss various issues regarding requirements and problems.

Chapter 5

5.1 Scope creep can be defined as a continuous increase in the scope of a project that can significantly increase project cost. Explain how a proper model of the system context can help prevent scope creep

At an early stage in the specification of a system, you should decide on the system boundaries, that is, on what is and is not part of the system being developed. This involves working with system stakeholders to decide what functionality should be included in the system and what processing and operations should be carried out in the system's operational environment. You may decide that automated support for some business processes should be implemented in the software being developed but that other business processes should be manual or supported by different systems. You should look at possible overlaps in functionality with existing systems and decide where new functionality should be implemented. These decisions should be made early in the process to limit the system costs and the time needed for understanding the system requirements and design.

Once some decisions on the boundaries of the system have been made, part of the analysis activity is the definition of that context and the dependencies that a system has on its environment. Producing a simple architectural model is the first step in this activity.

A context model normally show that the environment includes several other automated systems.
UML activity diagrams may be used to show the business processes in which systems are used. UML activity diagrams show the activities in a process and the flow of control from one activity to another.

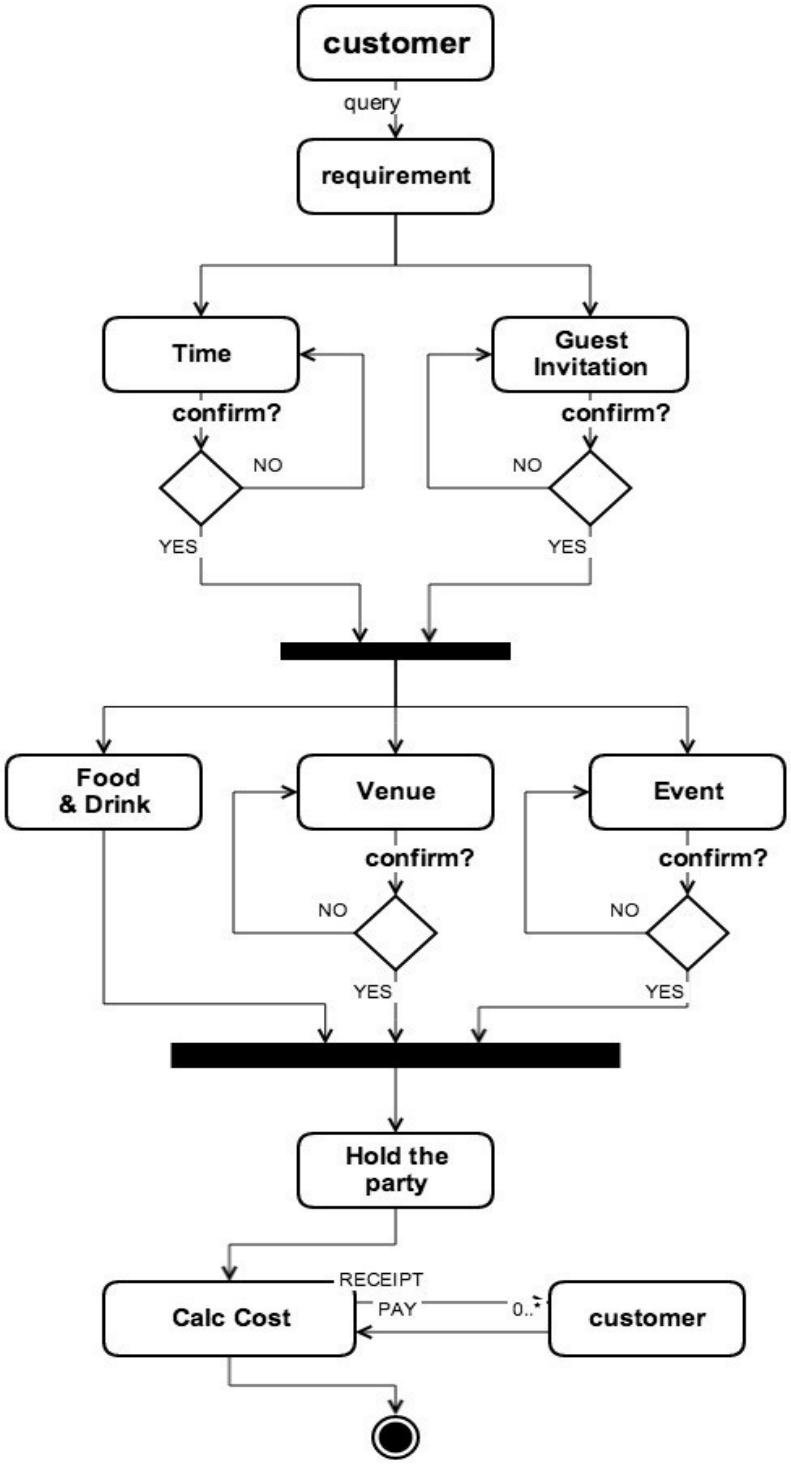
5.2 The way in which a system boundary is defined and an appropriate context model is created may have serious implications on the complexity and cost of a project. Give two examples where this may be applicable

Social and organisational concerns may mean that the position of a system boundary may be determined by nontechnical factors. For example, as system boundary may be deliberately positioned so that the complete analysis process can be carried out on one site; it may be chosen so that a particularly difficult manager need not be consulted; and it may be positioned so that the system costs is increased and the system development division must therefore expand to design and implement the system.

Once some decisions on the boundaries of the system have been made, part of the analysis activity is the definition of that context and the dependencies that a system has on its environment.

The Mentcare patient information system is intended to manage information about patients attending mental health clinics and the treatments that have been prescribed. In developing the specifications you have to decide whether the system should focus exclusively on collecting information about consultations or whether it should also collect personal patient information. The cost and complexity for the system to integrate to various other system will increase if this becomes an integrated Healthcare system reading data from various other systems and also providing data to those systems.

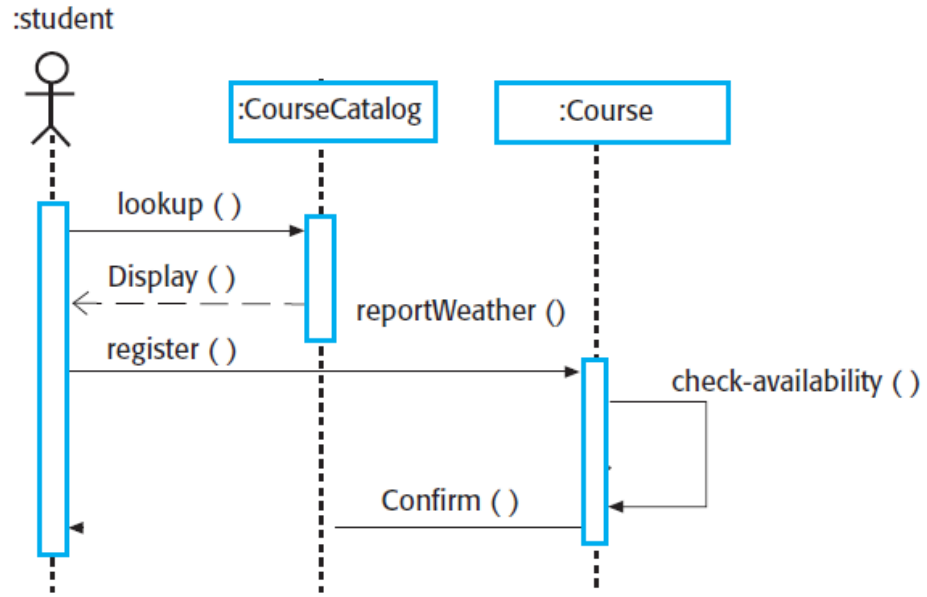
5.3 You have been asked to develop a system that will help with planning large-scale events and parties such as weddings, graduation celebrations and birthday parties. Using an activity diagram, model the process context for such a system that shows the activities involved in planning a party and the system elements that might be used at each stage.



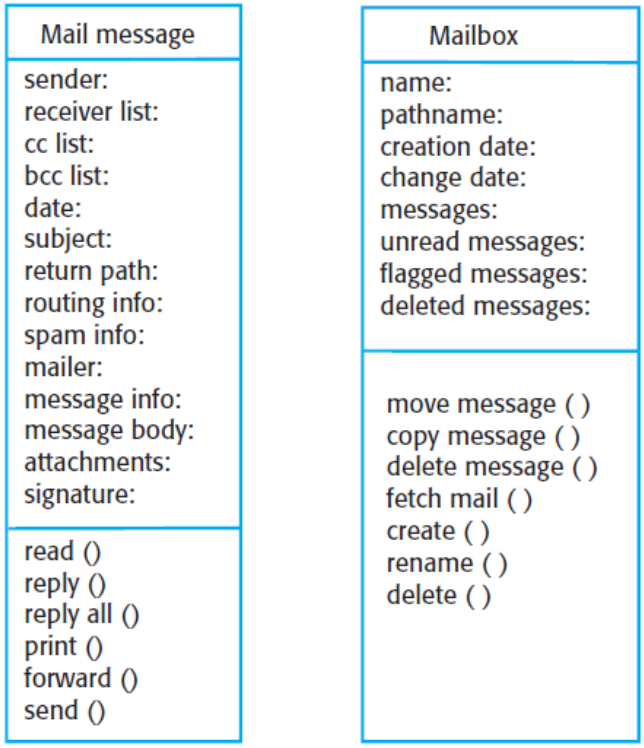
5.4 For the Mentcare system, propose a set of use cases that illustrates the interactions between a doctor, who sees patients and prescribes medicine and treatments, and the Mentcare system.

Actors	Doctor, Patient records system
Description	The doctor enters the patient's Unique Identification data into the Mentcare system The Patient records system transfer basic patient information to the doctor to view e.g. Personal information, diagnoses, previous recordings, treatment information The doctor records his session with the patient in the Mentcare system eg date, time, diagnoses The doctor records the treatment prescribed in the Mentcare system
Data	Patient's personal information, treatment summary
Stimulus	User command issued by the doctor
Response	Confirmation that the patient's record in the Mentcare system has been updated
Comments	The doctor must enter a valid unique Identification number to be able to access the patient's data The doctor must have appropriate security permissions to access the patient information

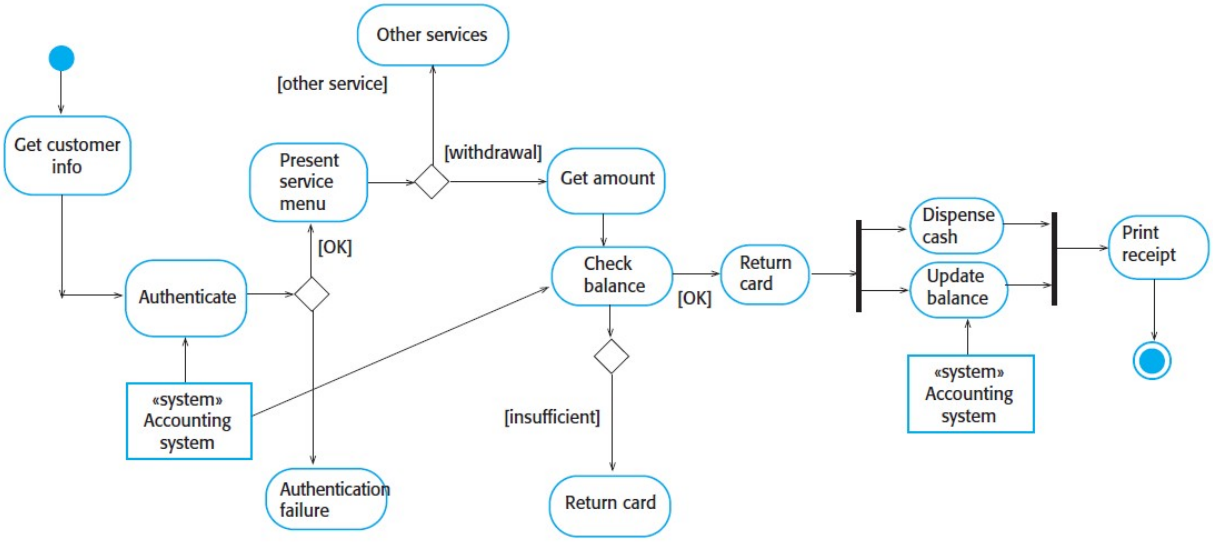
5.5 Develop a sequence diagram showing the interactions involved when a student registers for a course in a university. Courses may have limited enrolment, so the registration process must include checks that places are available. Assume that the student accesses and electronic course catalog to find out about available courses



5.6 Look carefully at how messages and mailboxes are represented in the email system that you use. Model the object classes that might be used in the system implementation to represent a mailbox and an email message

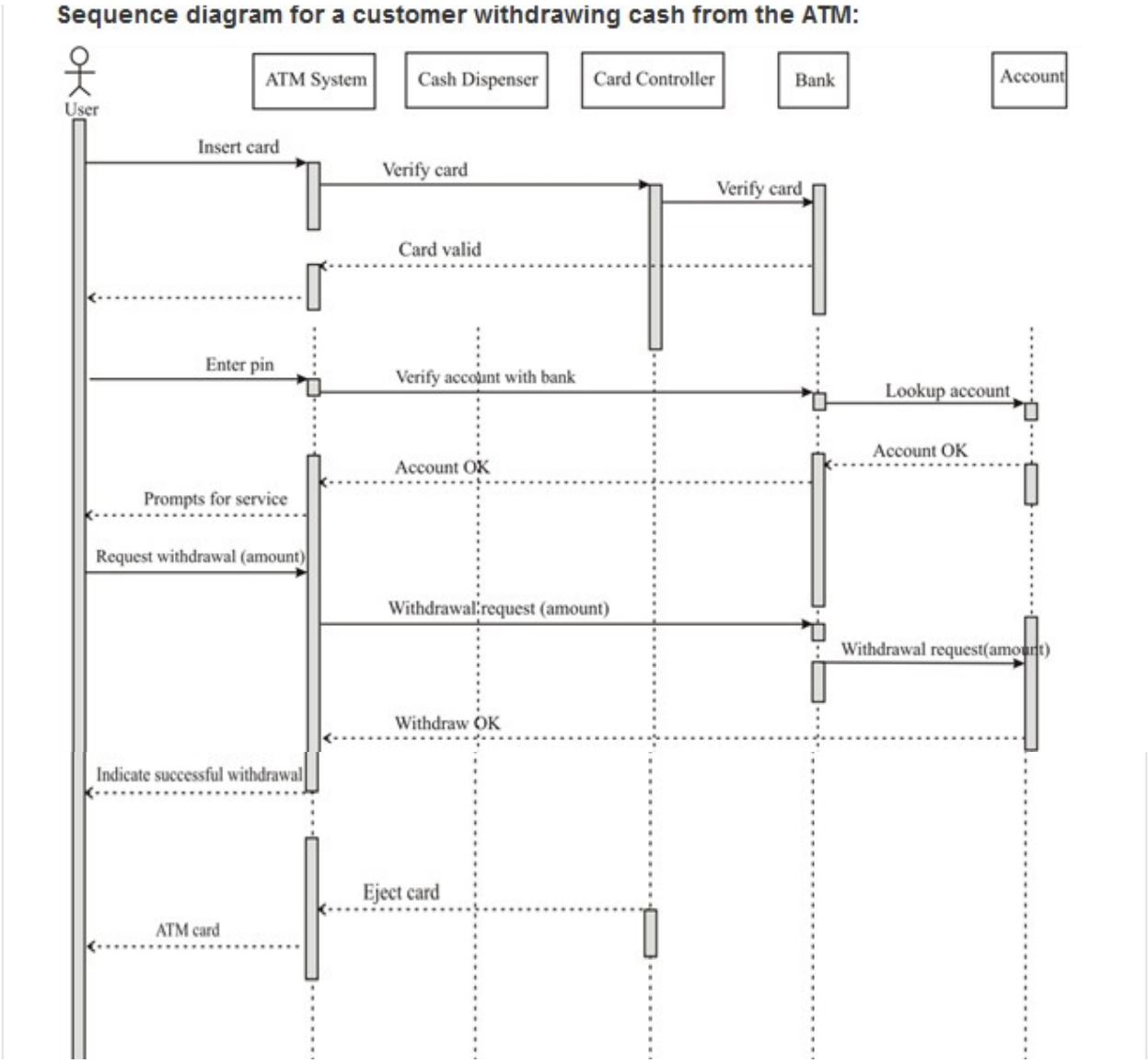


5.7 Based on your experience with a bank ATM, draw an activity diagram that models the data processing involved when a customer withdraws cash from the machine



5.8 Draw a sequence diagram for the same system. Explain why you might want to develop both activity and sequence diagrams when modelling the behaviour of a system.

Sequence diagram for a customer withdrawing cash from the ATM:



The models are used to explain the system in different perspectives

- An external perspective
- An interaction perspective
- A structural perspective
- A Behavioural perspective

Among the nine diagrams in UML, activity and sequence diagram are more useful for system modelling.

Activity diagram and sequence diagram are useful in modelling the behaviour of the system because of the following:

- Activity diagram(structural perspective)

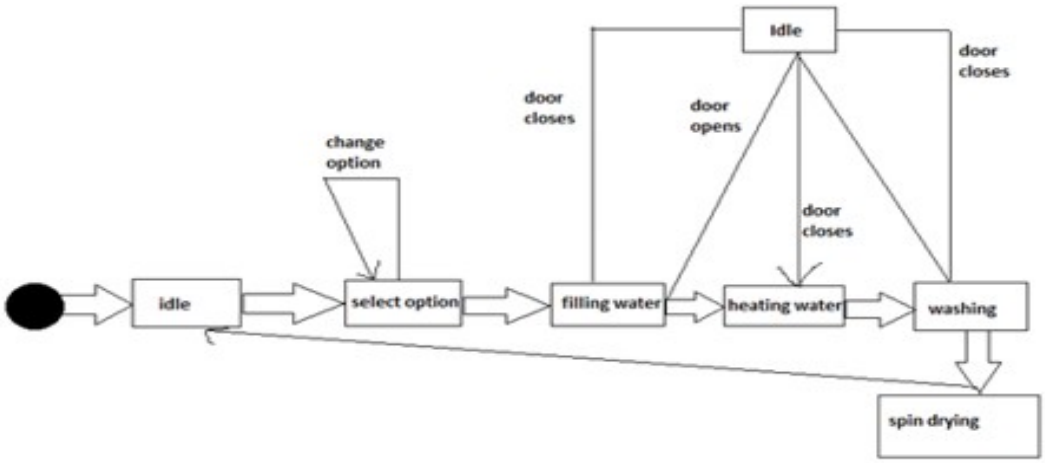
    - Process models are useful in revealing how the systems are being developed, is ues in business processes
    - Activity diagram are used to define business process models
    - Program flows of activities are illustrated by activity diagram
    - Activity diagram are used to describe the business processes which define the functionality of the business system.
- Sequence diagram (interaction perspective)
    - These diagrams are used to model interaction between the actors and the objects within the system
    - Sequence diagram are used to model the collaboration of objects based on time stamp
    - Sequence diagram is useful in gathering prerequisites, in a business level system implementation

5.9 Draw state diagrams of the control software for:

1. An automatic washing machine that has different programs for different types of clothes
2. The software for a DVS player
3. The control software for the camera on your mobile phone. Ignore the flash if you have one on your phone

A state diagram is used to show the connection between different activities of the system. State diagram show how the system reacts to internal and external events.

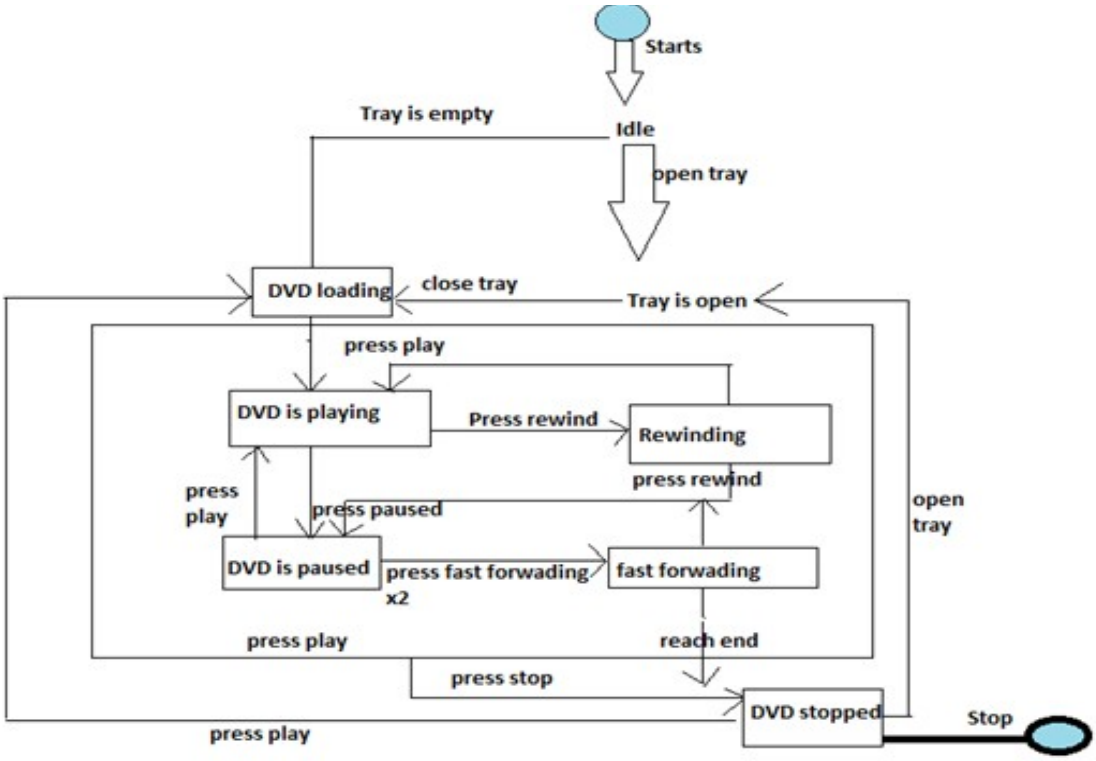
The state diagram for a washing machine is as given below:



Explanation

- Initially, washing machine is in idle state
- User select the option as per his requirements
- Fill machine with water
- After that washing machine starts water heating
- The machine starts washing the clothes
- The machine spin dry the clothes
- After the spin state, the machine goes back to the idle state
- When user opens the machine door during filling water, heating water, washing or spin drying, the machine goes back to its idle state

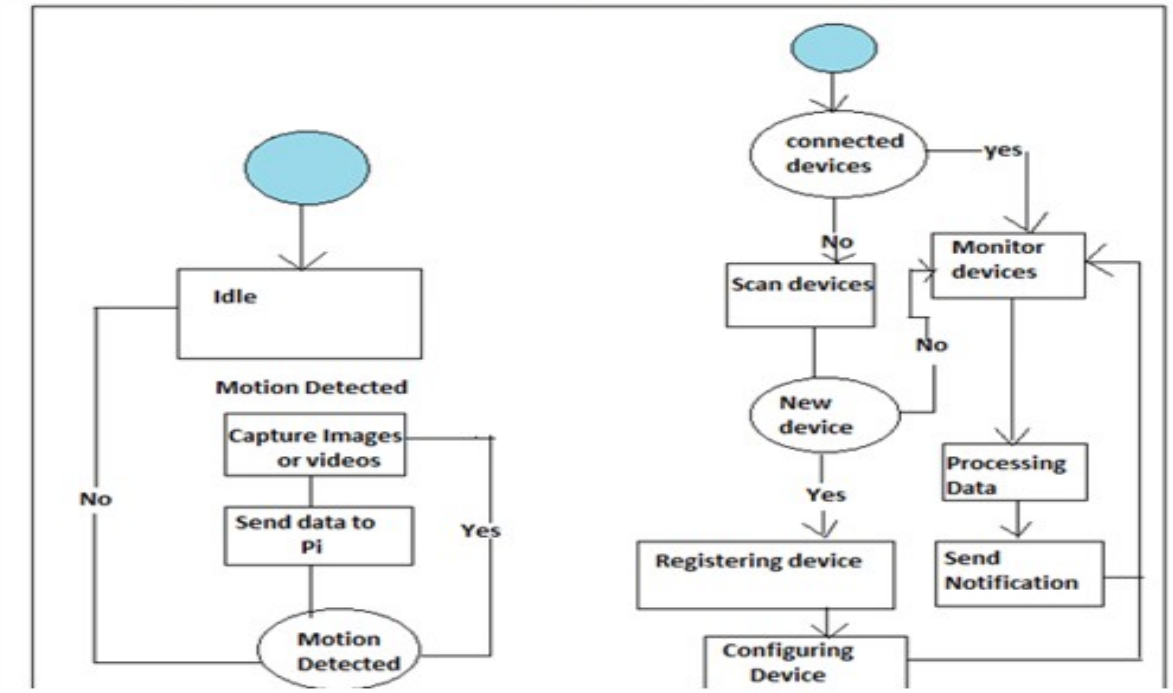
The state diagram for DVD player software:



Explanation:

- The DVD player is in idle state when user starts the DVD player
- User presses the open button to open the DVD tray
- User load the DVD onto the tray and press the close button
- User press the play button to view data
- User press the rewind button for play the same thing backwards
- Press pause button to stop laying the data, DVD player is in paused state
- Press play button again to resume the paused data
- Press the fast forwarding x2 button to cut the unwanted date
- The DVD reached the end point
- User presses the stop button to stop the DVD player
- The user wants to repeat the same process by pressing play button

The state diagram for camera software of any mobile phone is as given below:



Explanation:

- Initially, the camera is in its idle stage.
- Switch ON the mobile phone camera and capture the picture and record the video as per user's requirement.
- If the camera detects any motion, then, he/she can capture the picture and record the video.
- Captured picture and recorded video is saved into the Pi or memory device.
- If any motion is detected in recorded picture or video, then, capture the picture or record the video again.
- If any motion is not detected in picture or video, then, camera goes back to the idle stage.
- Now user tries to connect the camera to the System.
- The System monitors the camera if it is successfully connected.
- System process's the data which is stored in the pi or memory device.
- Now, the notification is sent by system to the user regarding data is saved or not.
- If the system is not able to detect the device, then, the system again scans for the device.
- The system finds a new device.
- Registered the new device on the System.
- After this process, the system again monitors the device.
- System processing the data which is stored in the pi or memory device.
- System send the notification to the user, data is saved or not into the device.
- Finally, the system goes to idle stage.

5.10 In principle, it is possible to generate working programs from a high-level model without manual intervention when using model-driven architectures. Discuss some of the current challenges that stand in the way of the existence of completely automated translation tools.

Model-driven architecture is a model-focuses approach to software design and implementation that uses a subset of UML models to describe a system. Here, models at different level of abstraction are created. From a high-level, platform independent model, it is possible, in principle, to generate a working program without manual intervention.

- A particularly difficult problem for automated model transformation is the need to link the concepts used in different CIMS.
- Off-the-shelf tool support is not availabe, so when MDA is introduced into an organisation, special-purpose translators may have to be created to make use of the facilities available in the local environment. Companies do not want to develop or maintain their own tools or to rely on small software companies for tool development.
- It doesn not always follow tha the abstractions that are useful for discussions are the right abstractions for implementation. You may decide to use a completely different implementation approach that is based on the reuse of off-the-shelf applciation systems.
- For most complex systems, implementation is not the major problem – requirements engineering, security and dependability, integration with legacy systems and testing are all more significant. Consequently the gains from the use of MDA are limited.
- The arguments for platform indepence are only valid for large, long-lifetime systems, where the platforms become obsolete during a system's lifetime. For software products and information systems that are developed fo rstandard platforms, such as Windows and Linux, the savings from the use of MDA are likely to be outweighed by the costs of its introduciton and tooling
- The widespread adoption of agile methods over the same period that MDA was evolving has diverted attention away from model-driven approaches.

Chapter 6

6.1 When describing a system, explain why you may have to start the design of the system architecture before the requirements specification is complete

You may have to design the system architecture before the requirements specification is complete because the architecture has a significant impact on the non-functional requirements and can also influence the functional requirements as well. Specifically, in order to demonstrate to stakeholders that an application will meet its performance requirements a project manager or system architect may have to show how the architecture will aid in accomplishing this goal. According to Sommerville the components affect the requirements and therefore an architecture that explains the components and their relationships may aid in the determination of the requirements.

An early stage of an agile development process should focus on designing an overall system architecture. Incremental development of architectures is not usually successful. Refactoring components in response to changes is usually relatively easy. However, refactoring the system architecture is expensive because

you may need to modify most system components to adapt them to the architectural changes.

The architecture may have to be designed before specifications are written to provide a means of structuring the specification and developing different subsystem specifications concurrently, to allow manufacture of hardware by subcontractors and to provide a model for system costing.

When describing a system, you may have to design the system architecture before the requirements specification is complete you need to model the system architecture in order to identify sub-systems and to associate requirements with each sub-system

The main reasons as to why System architecture must be designed before the requirements is that specifications is to provide the following.
¬ Means for structuring.
¬ A model for estimating system costing.
¬ To allow manufacture hardware by sub-contractors.
Since it is very complex to write specifications and formulate the whole system, the system is usually divided into simpler subsystems and this saves the developer from the hassle of defining specifications and putting them to their respective subsystems

6.2 You have been asked to prepare and deliver a presentation to a nontechnical manager to justify the hiring of a system architect for a new project. Write a list of bullet points setting out the key points in your presentation in which you explain the importance of software architecture

Software architecture is important because it affects the performance, robustness, distributability and maintainability of a system. Individual components implement the functional system requirements, but the dominant influence on the non-functional system characteristics is the system's architecture.

Three advantages of System architecture:

- Stakeholder communication: The architecture is a high-level presentation of the system that may be used as a focus for discussion by a range of different stakeholders.
- System analysis: making the system architecture explicit at an early stage in the system development requires some analysis. Architectural design decisions have a profound effect on whether or not the system can meet critical requirements such as performance, reliability and maintainability.
- Large-scale reuse: An architectural model is a compact, manageable description of how a system is organised and how the components interoperate. The system architecture is often the same for systems with similar requirements and so can support large-scale software reuse.

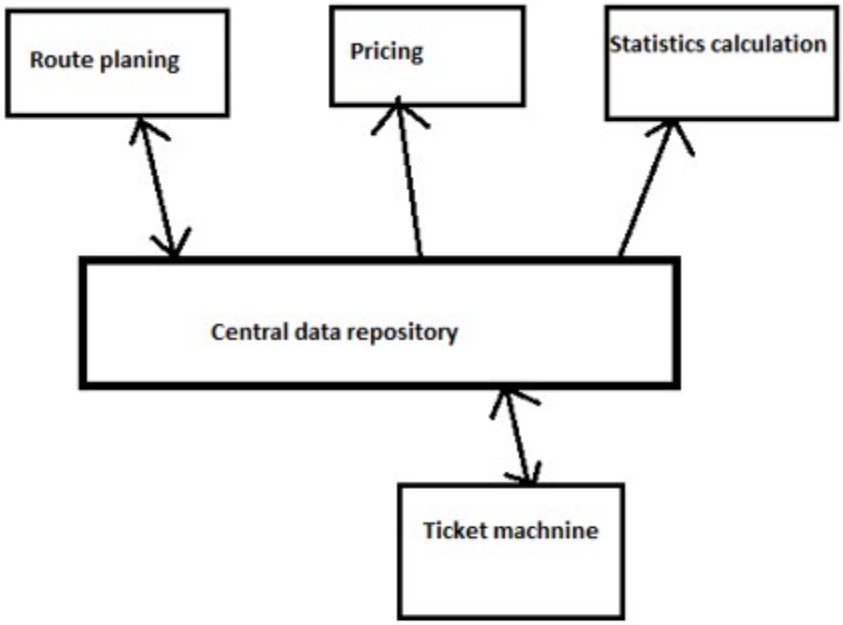
6.3 Performance and security may pose to be conflicting non-functional requirements when architecting software systems. Make an argument in support of this statement

Having a system that has both Performance and security requirements are the most important non-functional requirements is a design conflict because Performance means that a few relatively large components is used rather than small, finer-grain components. Using large components reduces the number of component communications, as most of the interactions between related system features take place within a component. But the system that has strong security would use a layered structure for the architecture, with the most critical assets protected in the innermost layers and a high level of security validation applied to these layers, which is not applicable when only one component is used and the finer-grain components bypassed.

6.4 Draw diagrams showing a conceptual view and a process view of the architectures of the following systems:

1. A ticket machine used by passengers at a railway station
2. A computer-controlled video conferencing system that allows video, audio and computer data to be visible to several participants at the same time
3. A robot floor-cleaner that is intended to clean relatively clear spaces such as corridors. The cleaner must be able to sense walls and other obstructions.

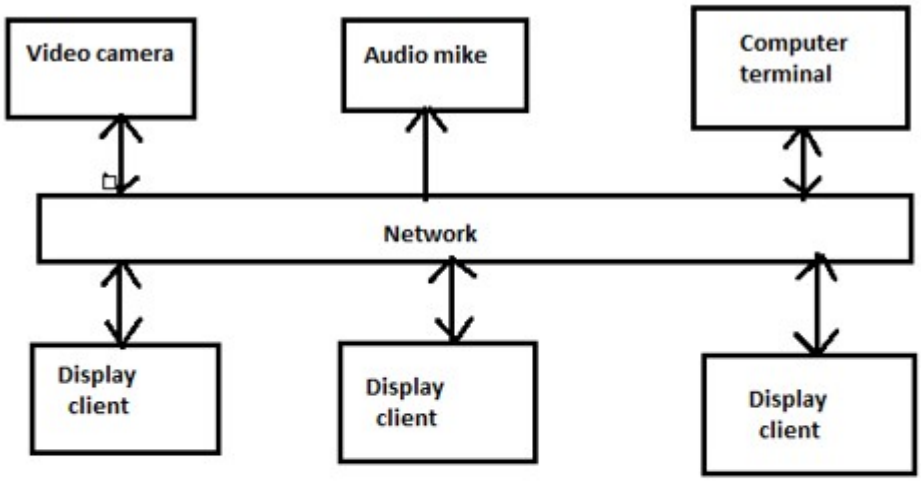
The system architecture for a ticket machine at a railway station is as given below:



Explanation:

- When a user books a ticket, he/she visits the ticket machine.
- The user uses the central data repository for booking the ticket.
- After using the central data repository, user checks the route of desire destination. Then the user checks the price for the ticket of his/her desire destination.
- The exact value of traveling is calculated.

The system architecture for a video conferencing system is as given below:



Explanation:

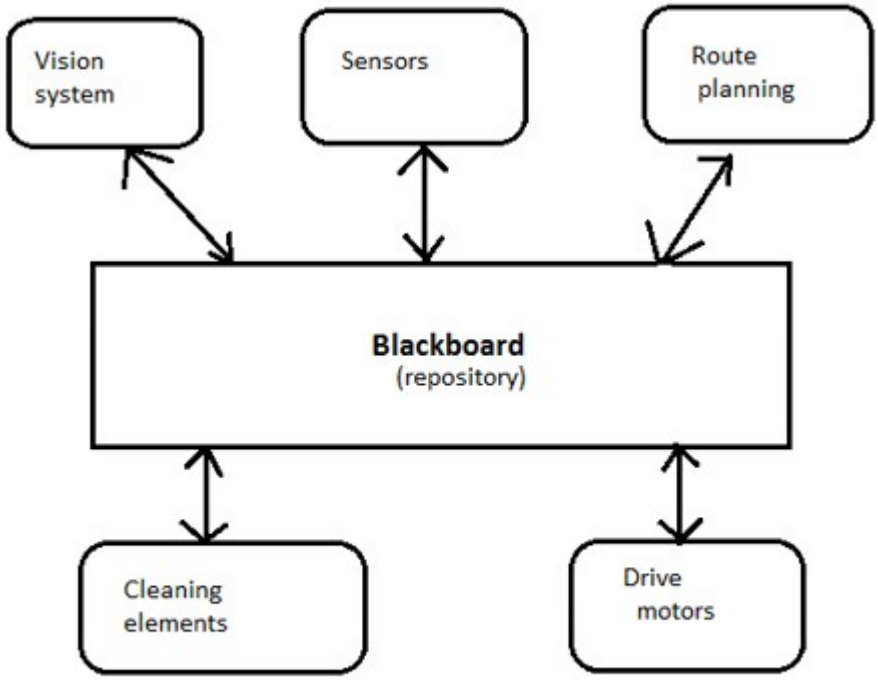
• Initially, for starting the video conferencing user's needs the video camera, Audio mike and computer terminal on both sides.

• Video camera and audio mike is required to both ends of the users to see each other and to listen their voice.

• A computer terminal is required to connect both video camera and audio mike.

• A display is required to the both end of the users to see each other.

The system architecture for a robot floor cleaner system is as given below:



Explanation:

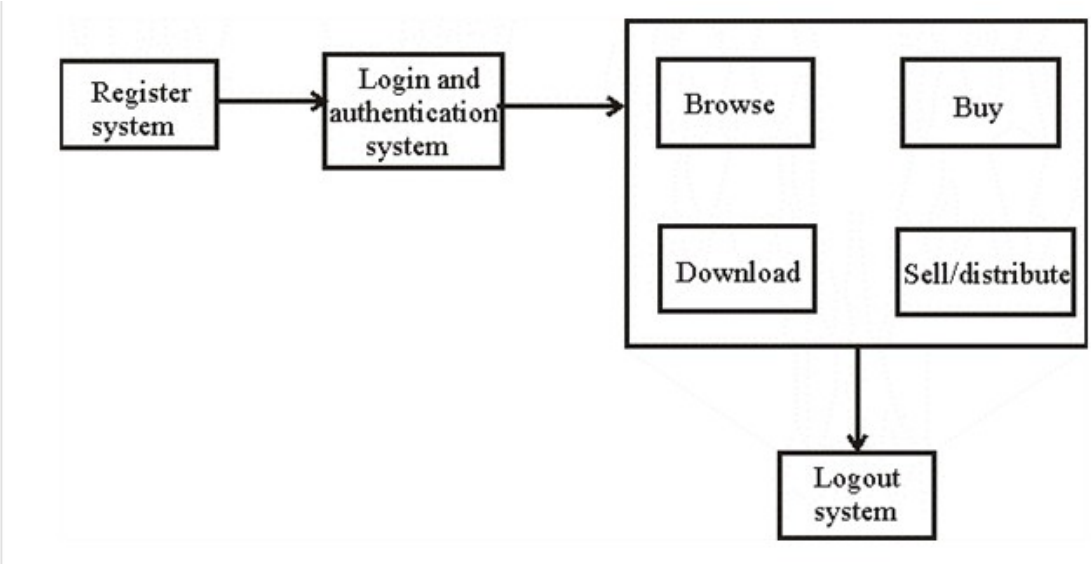
- Initially, user requires a robot for cleaning the floor. User controls the robot through a blackboard (repository).
- When a user gives the start command to the robot, Robot is going to the vision system (where the robot sees the path where it works).
- After that the robot on the sensor to find out the route planning where the robot works.
- Robot collect the cleaning element for clean the floor.
- Robot stars floor cleaning and this all process of cleaning is monitored by the user on blackboard (repository).

6.5 A software sysem will be built to allow drones to autonomously herd cattle in farms. These drones can be remotely controlled by human operators. Explain how multiple architectural patterns can fit toghether to help build this kind of system

The model manages fundamental behaviors and data of the application. It can respond to requests for information, respond to instructions to change the state of its information, and even to notify observers in event-driven systems when information changes. This could be a database, or any number of data structures

or storage systems. In short, it is the data and data-management of the application.
The view effectively provides the user interface element of the application. It'll render data from the model into a form that is suitable for the user interface.
The controller receives user input and makes calls to model objects and the view to perform appropriate actions.
All in all, these three components work together to create the three basic components of MVC.

6.6 Suggest an architecture for a system (such as iTunes) that is used to sell and distribute music on the internet. What architectural patterns are the basis for your proposed architecture?

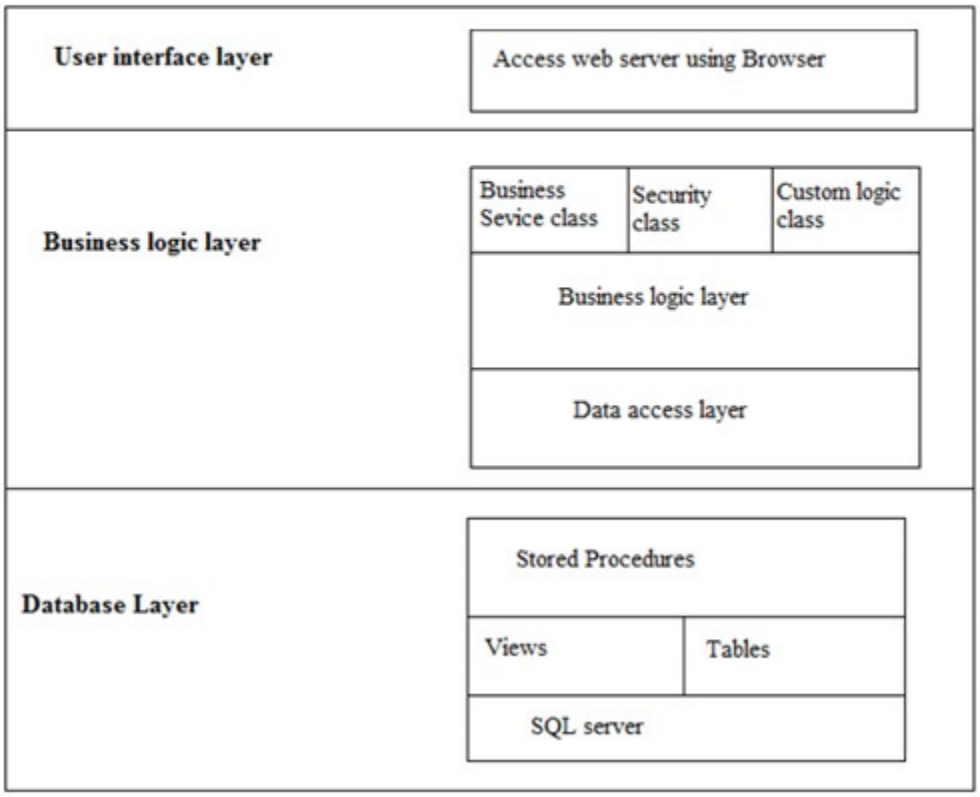


Client server model is the appropriate architectural pattern for such kind of system. iTunes stores all the music they sell in a database where the client can search these tracks by artist name, genre, etc. all via web based interface. Also, tracks can be downloaded and paid accordingly. Then, the server manages the music ordering also via web based interface.

6.7 An information system is to be developed to maintain information about assets owned by a utility company such as buildings, vehicles, and equipment. It is intended that this will be updatable by staff working in the field using mobile device as new asset information becomes available. The company has several existing asset databases that should be integrated through this system. Design a layered architecture for this asset management system based on the generic information system architecture shown in figure 6.18

Asset management system

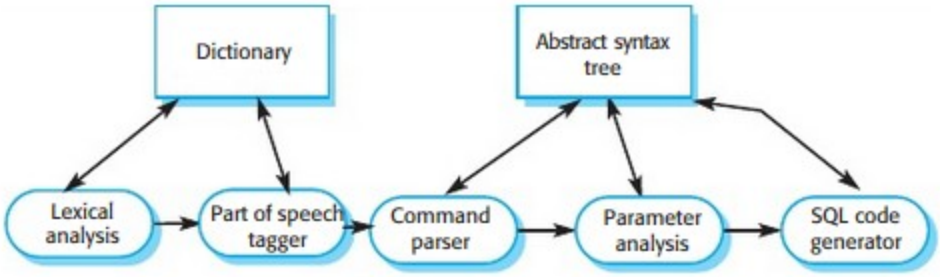
An asset management system has many layers which are related to security, set of rules, etc. The diagram to show the layered architecture of asset management system is as given below:



There are three types of layers for architecture of an asset management system:

- User interface layer: user can access web services using the browser
- Business logic layer: user has some options. User goes to the business logic layer and access the data on the data access layer
- Database layer: user sees the stored procedures, different views, table and SQL server

6.8 Using the generic model of a language processing system presented here design the architecture of a system that accepts natural language commands and translates these into database queries in a language such as SQL.



6.9 Using the basic model of an information system, as presented in Figure 6.18, suggest the components that might be part of an information system that allows users to view box office events, available tickets and prices, and so eventually buy tickets.

Web browser

Login	Credential Authentication	Data validation	Event schedule manager

Event info manager	Security management	Data exchange

Transaction management Event database

The top layer implements the user interface. The component responsible for this implementation is the web browser.

The second layer includes the components to allow users to log into the system, verify their credentials and data entered and obtain event schedules and availability.

The third layer contains components that implement system security, maintain even information and exchange information between databases and provide it to the users.

The fourth layer contains database and transaction management components.

6.10 Should there be a separate profession of 'software architect' whose role is to work independently with a customer to design the software system architecture? A separate software company would then implement the system. What might be the difficulties of establishing such a profession?

The difficulties that may arise by the establishment of such a profession are:

1. Difficulty in the implementation
2. The company that would implement the system may find difficulty in understanding the architecture
3. Incompatibility of the architecture with the developing system
4. Architecture may not satisfy all the technical requirements
5. Re-work or redesign is needed sometimes, due to incompatibility or change in technical requirements

Chapter 7

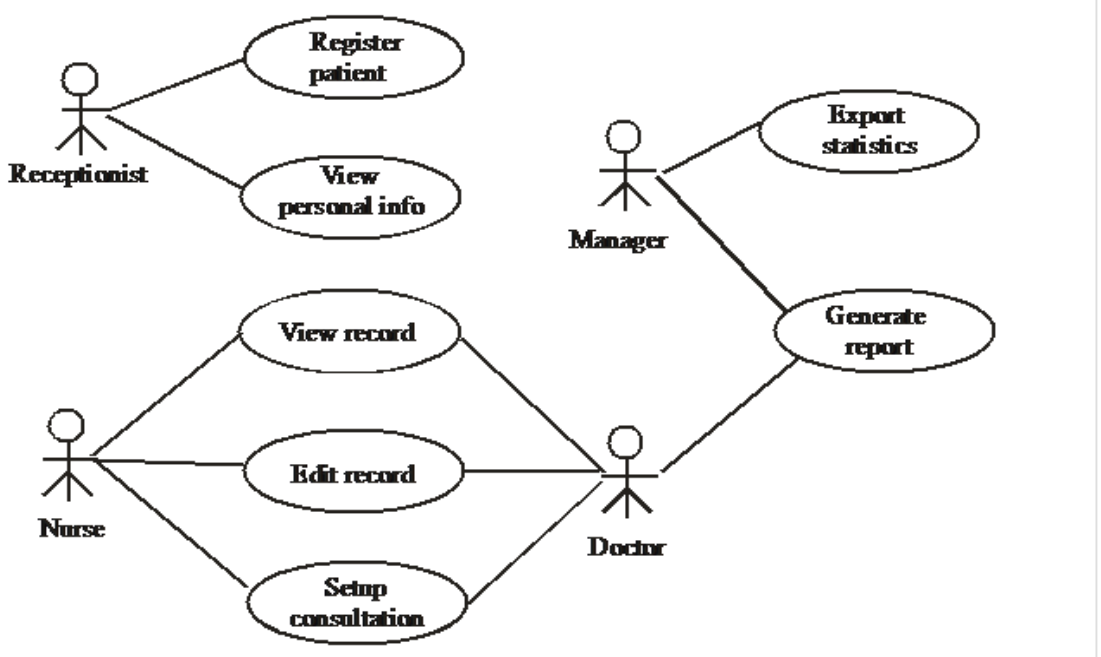
7.1 Using the structured notation shown in Figure 7.3, specify the weather station use cases for Report status and Reconfigure. You should make reasonable assumptions about the functionality that is required here.

System: Weather station
Use case: Report status
Actors: Weather information system, weather station
Data: The weather station sends a status update to the weather information system giving information about the status of its instruments, computers and power supply.
Stimulus: The weather information system establishes a satellite link with the weather station and requests status information.

Response: A status summary is uploaded to the weather information system
Comments: System status is usually requested at the same time as the weather report.

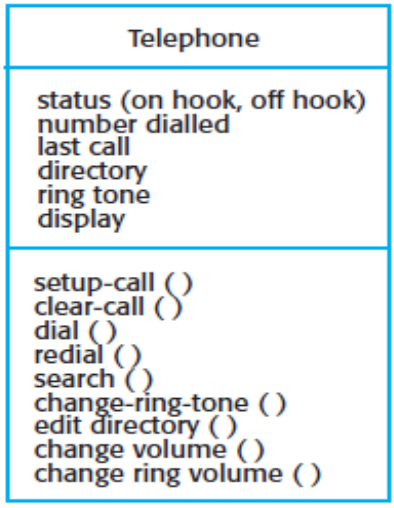
System: Weather station
Use case: Reconfigure
Actors: Weather information system, weather station
Data: The weather information station sends a reconfiguration command to the weather station. This places it into remote control mode where further commands may be sent from the remote system to update the weather station software.
Stimulus: A command from the weather information system.
Response: Confirmation that the system is in remote control mode
Comments: Used occasionally when software updates have to be installed.

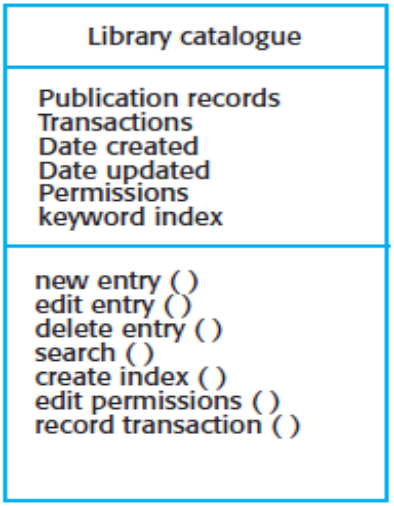
7.2 Assume that the mentcare system is being developed using an object-oriented approach. Draw a use case diagram showing at least six possible use cases for this system

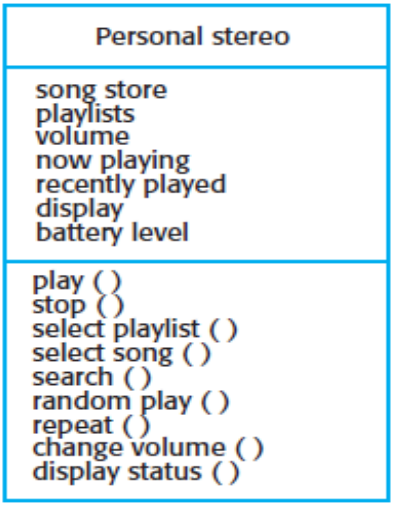


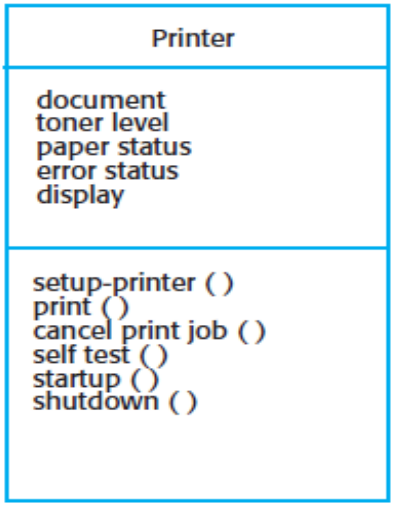
7.3 Using the UML graphical notation for object classes, design the following object classes, identifying attributes and operations. Use your own experience to decide on the attributes and operations that should be associated with these objects.

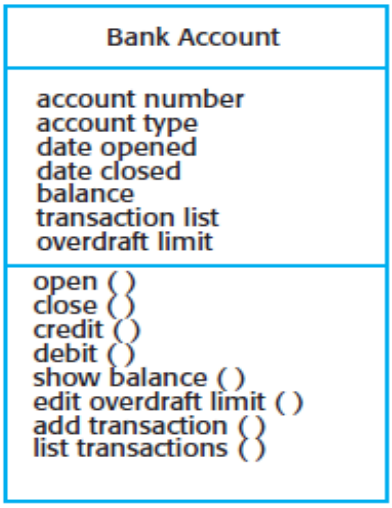
1. a telephone
2. a printer for a personal computer
3. a personal stereo system
4. a bank account
5. a library catalogue



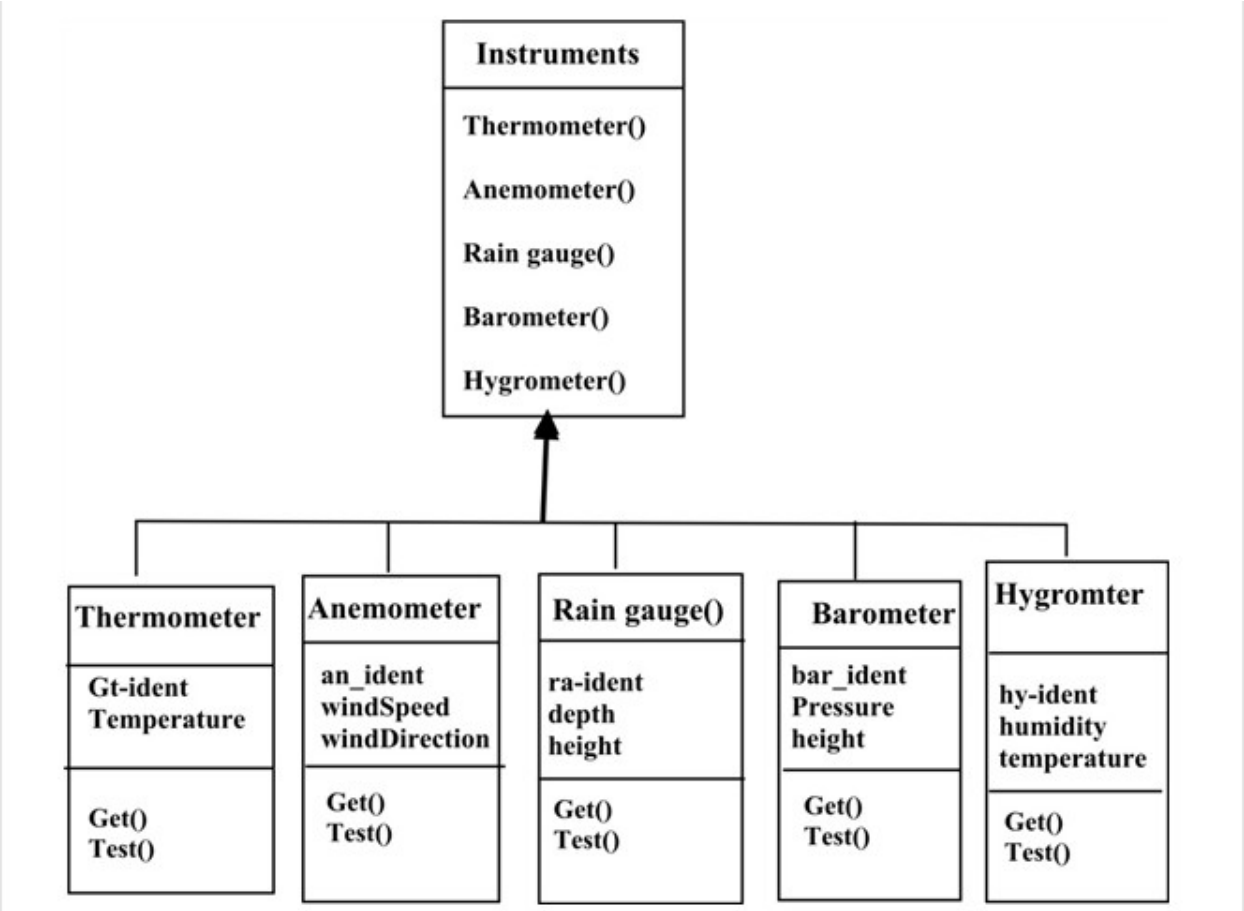








7.4 A shape can be classified into 2-D and 3-D. Design an inheritance hierarchy that will include different kinds of 2-D and 3-D shapes. Make sure you identify at least five other classes of shapes.



1. To calculate the weather report, different types of instruments are provided.

2. In the above figure, **Instruments** object is taken as superclass.

3. The domain objects are inherited from the superclass. The domain objects are Thermometer, Anemometer, Rain gauge, Barometer, and Hygrometer.

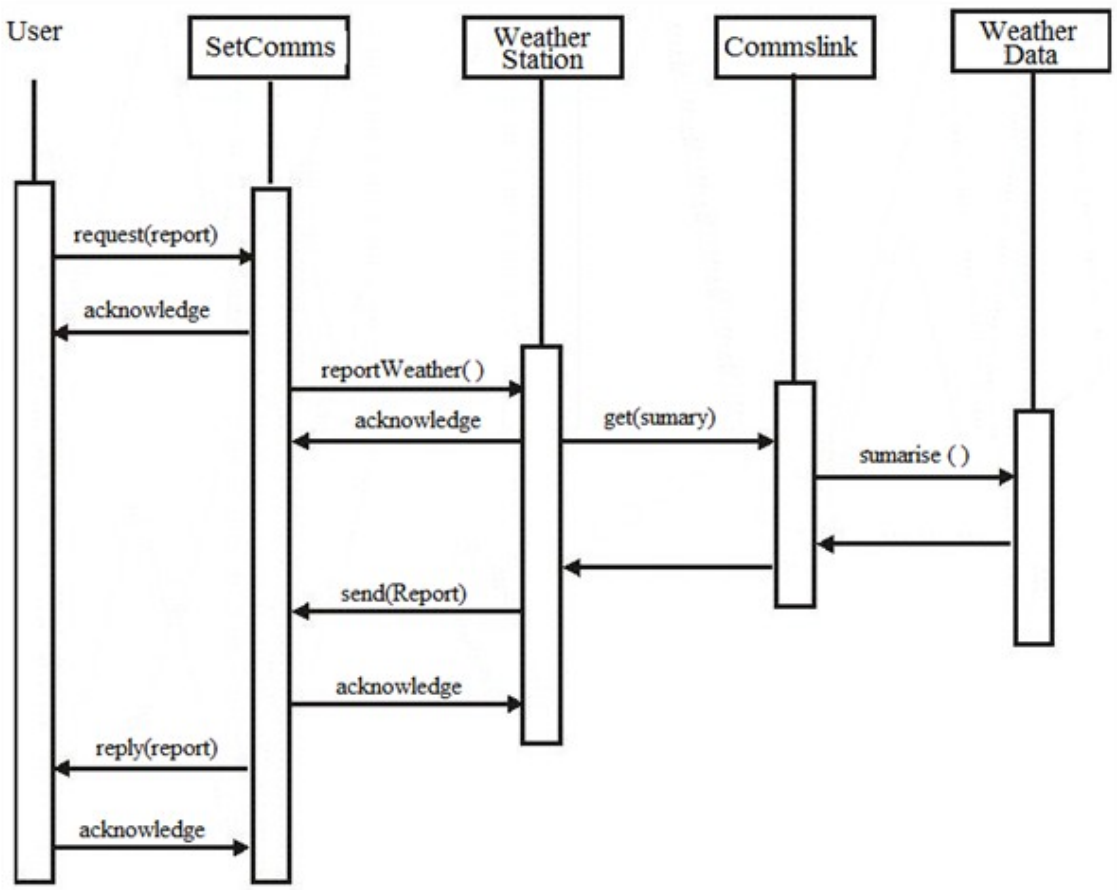
4. In **Thermometer** object have identifiers like unique identifier **gt-ident** and amount of **temperature** and the function are get() and test() are used to calculate and test the report.

5. In **Anemometer** object have identifiers like unique identifier **an-ident** and amount of **windSpeed** and **windDirection** and the function are get() and test() are used to calculate and test the report.

6. In **Rain gauage** object have identifiers like unique identifier **ra-ident** and amount of **depth** and **height** of the wind and the function are get() and test() are used to calculate and test the report.

7. In **Barometer** object have identifiers like unique identifier **bar-ident** and amount of **pressure** and **height** of the wind and the function are get() and test() are usedto calculate and test the report

7.5 Develop the design of the weather station to show the interaction between the data collection subsystem and the instruments that collect weather data. Use sequence diagrams to show this interaction


User
SetComms
Weather Station
Commslink
Weather Data
request(report)
acknowledge
reportWeather( )
acknowledge
get(sumary)
sumarise ( )
send(Report)
acknowledge
reply(report)
acknowledge

7.6 Identify possible object in the following systems and develop an object-oriented design for them. You may make any reasonable assumptions about the systems when deriving the design.

1. A group diary and time management system is intended to support the timetabling of meetings and appointments across a group of co-workers. When an appointment is to be made that involves a number of people, the system finds a common slot in each of their diaries and arranges the appointment for that time. If no common slots are available, it interacts with the user to rearrange his personal diary to make room for the appointment
2. A filling station is to be set up for fully automated operation. Drivers swipe their credit card through a reader connected to the pump; the card is verified by communication with a credit company computer, and a fuel limit is established. The driver may then take the fuel required. When fuel delivery is complete and the pump hose is returned to its holster, the driver's credit card account is debited with the cost of the fuel taken. The credit card is returned after debiting. If the card is invalid, the pump returns it before fuel is dispensed.

a.

Possible principal objects of diary and time management system with their operations and attributes. Here there is a single diary object with different operations for group appointments and personal appointments.

Object	Attributes	Operations
Dairy	year Weeks_of_year Time_slots Access_permissions	make_appointment cancle_appointment move_appointment make_group_appointment Fing_free_slot Reserve_slots Book_slots Free_slots Display_diary Check slot status
Appointment	time Duration Place Participants Reason	
User	Dairy	Check_time_slot

Card_reader	Card_number Card_type Card_status Credit_limit	read_card check_status print_receipt
Fuel_tank	current_fuel_level	add_fuel Remive_fuel
Communication_system	number_dialled Credit_limit	send_card_number return_card_status
System_controller	Card_number Card_type Max_delivery Price_table Fuel_delivered	
Price_table	fuel_prices	lookup Amend_price

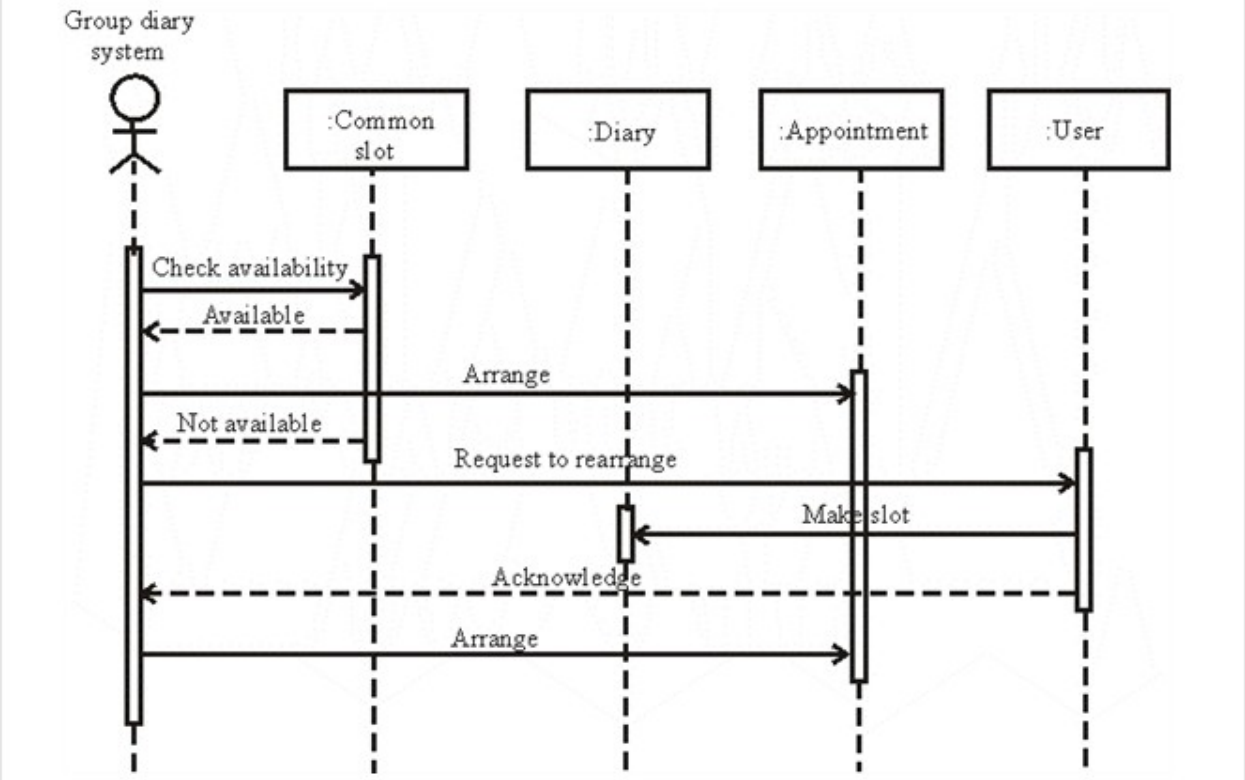
D.

Complete design about Gas filling station:

Here operations and attributes are associated with each object in the fuel tank system and provided a partial description of the system controller.

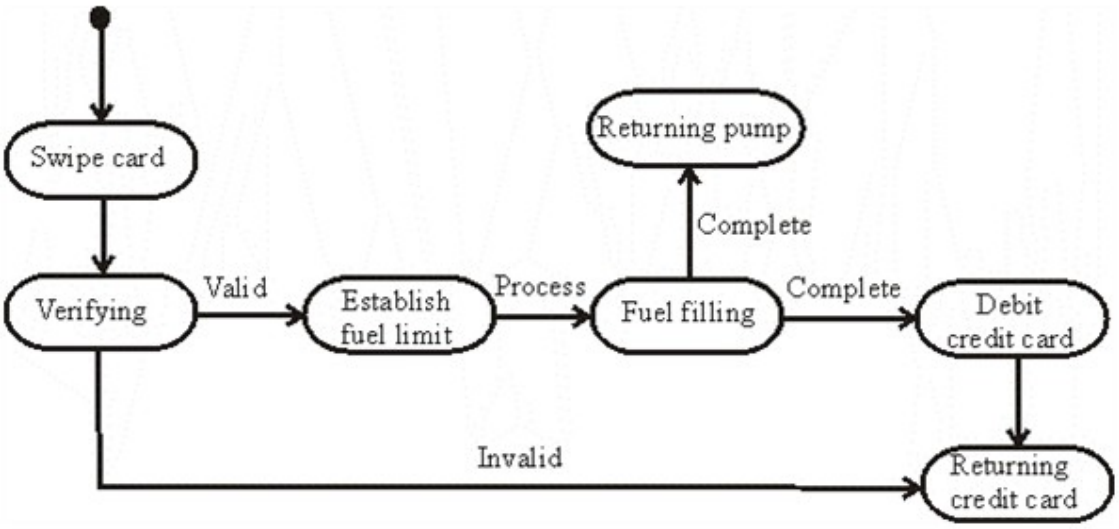
Object	Attributes	Operations
Pump	fuel_dispensed Price Hose_status Trigger_status Fuel_type	active deactivate deliver_fuel stick_update
Card_reader	Card_number Card_type Card_status Credit_limit	read_card check_status print_receipt
Fuel_tank	current_fuel_level	add_fuel Remive_fuel
Communication_system	number_dialled Credit_limit	send_card_number return_card_status
System_controller	Card_number Card_type Max_delivery Price_table Fuel_delivered	
Price_table	fuel_prices	lookup Amend_price

7.7 Draw a sequence diagram showing the interactions of objects in a group dairy system when a group of people are arranging a meeting



1. The group diary system checks for an availability of common slot.
2. If available, it arranges an appointment.
3. If a common slot is not available, it requests to arrange the appointments of the members.
4. When the users make a slot and intimates, the system arranges an appointment.

7.8 Draw a UML state diagram showing the possible state changes in either the group diary or the filling station system



The filling station system performs the actions in various states as shown:

1. When the swipe card state is completed, the card details are verified in the verifications state.

2. If the credentials are valid then the fuel limit is established and the fuel filling state is processed.

3. After the fuel filling state, the returning pump state occurs and the card is debited.

4. When the card is debited or if the credentials are not matched, the card returning state occurs.

7.9 When code is integrated into a larger system, problems may surface. Explain how configuration management can be useful when handling such problems

The aims of configuration management is to ensure that

(a) changes made by different system developers do not interfere with each other and

(b) it is always possible to create a specific version of a system.

Without configuration management it is easy to lose track of the changes that each developer makes to code and for changes made by one programmer to

overwrite changes made by another programmer. For example, one programmer may change a component to improve its performance whilst another may correct a bug in the functionality of the component. Without CM, whoever writes the component last to the shared component store will overwrite and so lose the previous component changes.
Furthermore, systems are usually composed of multiple components, each of which exists in multiple versions, where each version as a specific purpose. For example, there may be a versions of a system for different platforms such as Windows, Linux and MacOS. These versions have some specific components and some shared components and it is potentially error prone if these versions are assembled without CM tool support. It is very easy to include the wrong component in a version and this is likely to lead to subsequent software failure.
7.10 A small company has developed a specialized product that it configures specially for each customer. New customers usually have specific requirements to be incorporated into their system, and they pay for these to be developed. The company has an opportunity to bid for a new contract, which would more than double its customer base. The new customer also wishes to have some involvement in the configuration of the system. Explain why, in these circumstances, it might be a good idea for the company owning the software to make it open source.

The key benefits of open source are is that it opens up development to a wide range of developers and so accelerates the development and debugging of the product. Doubling the customer base places immense strains on a small company of they have to take on a lot of new staff and so going open source means that the costs of expansion are reduced.

In this case, because the product is specialized to the needs of different users, the company that own the software can still charge these users to make the changes to the system. Hence the loss in revenue from selling the software is compensated by the additional effort available to service more customers.
Furthermore, large companies are often reluctant to buy from small companies who may go out of business, To some extent, open source provides reassurance to customers that, even of the original owners of the software are unavailable, they can get access to the source code and hence continue to maintain their system.

Finally, open source may increase knowledge of the company's product and so attract more customers.

Chapter 8

8.1 Explain how the number of defects remaining in a program at the time of delivery affect product support

Testing is a process which shows a program does what is intended to do and to discover program defects before it is put into use. Testing cannot completely

validate that a system is fit for its intended purpose as this requires a detailed knowledge of what that purpose will be and exactly how the system will be used.

A program need not be completely free of defects before delivery if:

1. Remaining defects are minor defects that do not cause system corruption and which are transient i.e. which can be cleared when new data is input.
2. Remaining defects are such that they are recoverable and a recovery function that causes minimum user disruption is available.
3. The benefits to the customer's business from the system exceed the problems that might be caused by the remaining system defects.

Testing cannot completely validate that a system is fit for its intended purpose as this requires a detailed knowledge of what that purpose will be and exactly how the system will be used. As these details inevitably change between deciding to procure a system and deploying that system, the testing will be necessarily incomplete. In addition, it is practically impossible for all except trivial system to have a complete test set that covers all possible ways that the system is likely to be used.

8.2 Testing is meant to show that a program does what it is intended to do. Why may testers not always know what a program is intended for?

Testing is intended to show that a program does what it is intended to do and to discover program defects before it is put into use. Testing cannot demonstrate that the software is free of defects or that it will behave as specified in every circumstance. It is always possible that a test that you have overlooked, could discover further problems with the system.

Main problem here is whether or not the embedded user is ‘typical’ and can represent the interests of all system stakeholders. As stakeholders are consulted, the requirements might change.

8.3 Some people argue that developers should not be involved in testing their own code but that all testing should be the responsibility of a separate team. Give arguments for and against testing by the developers themselves.

Pro for Inside Team:

- The team understands the problems that their program is tackling, and thus understands the problems that it might run into.
- They can write tests that accurately reflect the programs' future challenges.
- An outsider team may try to verify the code, without fully understanding the initial problem.
- Testing becomes easy as they know the areas of code which need testing
- Saves time, as they need not learn about the code

Con for Inside Team:

- Depending on the team, and often pressured by time, they could write test units that insufficiently test the quality of the code, or are unable to think of challenges to their code beyond the solution that they've already tested.
- One may not easily identify their own errors
- Testing may be compromised so as to reduce their error rate
- Increases the effort of developers and may compromise with quality

The main concern of developer testing is – misunderstanding of requirements. If requirements are misunderstood by developer then no matter at what depth developer test the application, he will never find the error. The first place where the bug gets introduced will remain till the end, a developer will see it as functionality.

Developer always wants to see his code working properly. So he will test it to check if it's working correctly. But you know why tester will test the application? To make it fail in any way, and tester surely will test how an application is not working correctly. This is the main difference in developer testing and tester testing.

8.4. You have been asked to test a method called 'catWhiteSpace'in a 'Paragraph'object that, within the paragraph, replaces sequences of blank characters with a single blank character. Identify testing partitions for this example and derive a set of tests for the 'catWhiteSpace' method.

Testing partitions are

- Strings with only single blank characters
- Strings with sequences of blank characters in the middle of the string
- Strings with sequences of blank characters at the beginning/end of string

Examples of tests

The quick brown fox jumped over the lazy dog (only single blanks)

The quick brown fox jumped over the lazy dog (different numbers of blanks in the sequence)

The quick brown fox jumped over the lazy dog (1st blank is a sequence)

The quick brown fox jumped over the lazy dog (Last blank is a sequence)

The quick brown fox jumped over the lazy dog (2 blanks at beginning)

The quick brown fox jumped over the lazy dog (several blanks at beginning)

The quick brown fox jumped over the lazy dog (2 blanks at end)

The quick brown fox jumped over the lazy dog (several blanks at end)

8.5. What is regression testing? Explain how the use of automated tests and a testing framework such as JUnit simplifies regression testing.

Regression testing is the process of running tests for functionality that has already been implemented when new functionality is developed or the system is changed. Regression tests check that the system changes have not introduced problems into the previously implemented code.

Automated tests and a testing framework, such as JUnit, radically simplify regression testing as the entire test set can be run automatically each time a change is made.

The automated tests include their own checks that the test has been successful or otherwise so the costs of checking the success or otherwise of regression tests is low.

8.6. The Mentcare system is constructed by adapting an off-the-shelf information system. What do you think are the differences between testing such a system and testing software that is developed using an object-oriented language such as Java?

Differences between testing an off-the-shelf information system and object oriented software exists in their testing methodologies and various tests applied

- An application systems such as Mentcare must be tested by considering each requirement and every requirement carefully and by developing test cases. It should be tested that the applications works well in all the cases
- Object oriented software is tested in the perspective of classes, objects, their binding, dependency etc. whereas off-the-shelf information system applications must be tested in various perspectives such as requirements satisfaction, user acceptance, fault tolerance etc.

- The applications adapting an off-the –shelf information system are to be tested from environmental perspectives. For example if drug dosage is given as a wrong input, it affect the patient. All such conditions are to be tested thoroughly, whereas object oriented software may not result in such harmful effects.
- These testing process of the off-the-shelf systems must concentrate more on scenario testing, performance testing and stress testing. All these tests are to be performed because any error in the applications may affect more and may cause loss.

8.7. Write a scenario that could be used to help design tests for the wilderness weather station system.

A possible scenario for high-level testing of the weather station system is:
John is a meteorologist responsible for producing weather maps for the state of Minnesota. These maps are produced from automatically collected data using a weather mapping system and they show different data about the weather in Minnesota. John selects the area for which the map is to be produced, the time period of the map and requests that the map should be generated. While the map is being created, John runs a weather station check that examines all remotely collected weather station data and looks for gaps in that data – this would imply a problem with the remote weather station.

Scenario-based analysis of wilderness weather station system:
The weather station is composed of an independent subsystem that communicates by broad casting messages on a conman infrastructure. Weather stations collect data from a set of instruments that measure temperature and pressure, sunshine, rainfall, wind speed and wind direction.
The weather station includes a number of instruments that measure weather parameters such as the wind speed and direction, the ground and air temperatures, the barometric pressure and the rainfall over a 24-hour period. Each of these instruments is controlled by a software system that takes parameter readings periodically and manages the data collected from the instruments
In this objects, attributes and methods are identified. A weather station is a package of software controlled instruments which collect data. For designing a test, it is necessary to understand about the interactions between the system and its environment. Testing is performed by checking the data processing and

transmits ground thermometers, an anemometer, a wind vane, a barometer and a rain gauge.

8.8. What do you understand by the term ‘stress testing’? Suggest how you might stress test the Mentcare system.

Stress testing is where you deliberately increase the load on a system beyond its design limit to see how it copes with high loads. The system should degrade gracefully rather than collapse.
The Mentcare system has been designed as a client-server system with the possibility of downloading to a client. To stress test the system, you need to arrange for
(a) many different clinics to try and access the system at the same time and
(b) Large numbers of records to be added to the system.
This may involve using a simulation system to simulate multiple users.

Stress testing emphasizes availability and error handling under extremely heavy loads to ensure software does not crash due to insufficient resources. Software stress testing focuses on identified transactions to break transactions, which are heavily stressed during testing, even when a database has no load. The stress testing process loads concurrent users beyond normal system levels to find the system's weakest link.

The processes become slower and slower as they wait for the required data from other processes. Stress testing helps you discover when the degradation begins so that you can add checks to the system to rejects transactions beyond this point

Stress testing can be conducted through load testing tools, by defining a test case with a very high number of concurrent virtual users.

To stress test the Mentcare system, you need to arrange for as many different clinics to try and access the system at the same time and for large numbers of records to be added to the system. This may involve using a simulation system to simulate multiple users.

When testing the transaction processing capabilities of the Mentcare system, that is designed to process a certain amount of transactions per second, you may start

the testing with fewer than the designed transactions per second. You then gradually increase the load on the system beyond the designed amount of transactions per second until it is well beyond the maximum design load of the system and the system fails.

8.9. What are the benefits of involving users in release testing at an early stage in the testing process? Are there disadvantages in user involvement?

Advantages of involving users in release testing at an early stage:

- As the system release is for customers and users, they can easily identify the modifications needed
- Helps in getting the views of the users and the modifications needed from the users' perspective

Disadvantages of user involvement:

- May not get better feedback as the views and ideas of users vary from person to person
- May increase the effort of testing as the users may not have a clear view of the system
- Testing perspectives may sometimes deviate to some other view

8.10. A common approach to system testing is to test the system until the testing budget is exhausted and then deliver the system to customers. Discuss the ethics of this approach for systems that are delivered to external customers.

Ethics of testing the system until the testing budget is exhausted are:

- The testing may not be done satisfactorily
- Only few methods of testing are covered with the available budget
- Only few areas of code may be tested because of the running budget. So there may be some uncovered errors.
- The testing may not be completed in a satisfactory way by the time the budget exhausts
- Due to improper testing, errors may occur after delivering the project

- It increases the costs of error correction and maintenance
- Affect the reputation and goodwill of the product and vendor
- The customer may not be satisfied with the product as he may not get a quality product

Chapter 9

9.1 Explain how advances in technology can force a software subsystem to undergo change or run the risk of becoming useless

When a software system or subsystem is defined during the planning phase, the scope of the software system is defined - in terms i.e. what the software will be capable of doing or not doing. The scope also defines all the do's meaning all the activities and/or all the processes needs to be done to deliver the software, scope also creates defined boundaries which is easy in documentation and avoid cost and time spread-overs at the time of production.

With the arrival of new technology-change in technology the software might become slow/obsolete and at this point of time where the new technology is arriving the software also calls for improvement so that the software can incorporate the change in technology and adapt to the new environment of business. This is achieved by reusing the existing code which is also known as re-usability which also require some investment/cost.

Now at times it is impossible to re-use the existing code and therefore a new platform/ new software is required due to change in the overall environment of the business and also to incorporate the expanding scope of the business itself and then an requirement for new software system arises in which the new scope is defined, time and cost estimation is done, resource management is required, risk estimation and management is done, pert analysis is prepared etc is done.

Changes to one system may require consequent change to subsystems that are both above and below the changed system. Reasons for this are:

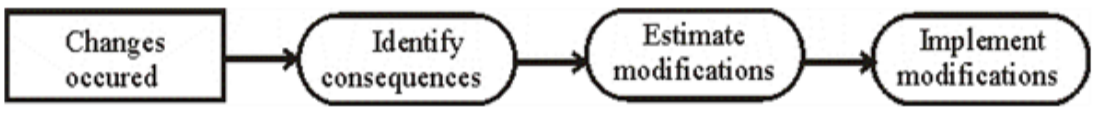
- Changing one layer in the system may introduce new facilities, and higher layers in the system may then be changed to take advantage of these facilities. Example: a new database introduces at the support software layer may include facilities to access the data through a web browser and business processes may be modified to take advantage of this facility.
- Changing the software may slow the system down so that new hardware is needed to improve the system performance. The increase in performance

from the new hardware may then mean that further software changes that were previously impractical become possible.

- It is often impossible to maintain hardware interfaces, especially if new hardware is introduced. This is particularly a problem in embedded systems where there is a tight coupling between software and hardware. Major changes to the application software may be required to make effective use of the new hardware.

9.2 From figure 9.4 you can see that impact analysis is an important sub-process in the software evolution process. Using a diagram, suggest what activities might be involved in change impact analysis

analysis.



Software evolution processes is vary and depending on the type of software being maintained. And, the change impact analysis includes the activities like identifying the consequences of the changes occurred, estimating the modifications need to be done to accomplish the changes and implementing the modifications

9.3 Explain why legacy systems should be thought of as socio-technical systems rather than simply software systems that were developed using old technology

Socio-technical systems – Systems that include technical systems but also operational processes and people who use and interact with the technical system. Socio-technical systems are governed by organisational policies and rules.

Socio-technical system characteristics

- Emergent properties – Properties of the system of a whole that depend on the system components and their relationships.
- Non-deterministic – They do not always produce the same output when presented with the same input because the system's behaviour is partially dependent on human operators.
- Complex relationships with organisational objectives – The extent to which the system supports organisational objectives does not just depend on the system itself.

Sociotechnical systems include hardware, software, libraries and other supporting software and business processes

1. System hardware: Legacy systems may have been written for hardware that is no longer available, that is expensive to maintain, and that may not be compatible with current organizational IT purchasing policies.
2. Support software: The legacy system may rely on a range of support software from the operating system and utilities provided by the hardware manufacturer through to the compilers used for system development. Again, these may be obsolete and no longer supported by their original providers.
3. Application software: The application system that provides the business services is usually made up of a number of application programs that have been developed at different times. Some of these programs will also be part of other application software systems.
4. Application data: These data are processed by the application system. In many legacy systems, an immense volume of data has accumulated over the lifetime of the system. This data may be inconsistent, may be duplicated in several files, and may be spread over a number of different databases.
5. Business processes: These processes are used in the business to achieve some business objective. An example of a business process in an insurance company would be issuing an insurance policy; in a manufacturing company, a business process would be accepting an order for products and setting up the associated manufacturing process. Business processes may be designed around a legacy system and constrained by the functionality that it provides.

6. Business policies and rules: These are definitions of how the business should be carried out and constraints on the business. Use of the legacy application system may be embedded in these policies and rules.

9.4 Some software subsystems are seen as "low quality, high business value". Discuss how those subsystems can be re-engineered with minimal impact on the operations of the organisation

These systems are making an important business contribution, so they cannot be scrapped. However, their low quality means that they are expensive to maintain. These systems should be re-engineered to improve their quality. They may be replaced, if suitable off-the-shelf systems are available.

To make subsystems easier to maintain, you can re-engineer these systems to improve their structure and understandability. Re-engineering may involve re-documenting the system, refactoring the system architecture, translating programs to a modern programming language or modifying and updating the structure and values of the system's data. The functionality of the software is not changed.

9.5 What are the strategic options for legacy system evolution? When would you normally replace all or part of a system rather than continue maintenance of the software

Legacy system is an operational system that has been designed, implemented and installed in a radically different environment. In critical business systems, it can use many legacy systems.

The most appropriate strategic options for evolving these systems are:

- Scrap the system completely. This option is performed when the system is ineffective to the business process or the business process has changed, leaving the system to be obsolete.
- Leave the system unchanged and continue regular maintenance. This option is chosen when the system is still required but is fairly stable and change requests happen rarely.

- Reengineer system to improve maintainability. This is performed when the system maintenance costs exceeds the cost of reengineering the system.
- Replace all or part of the system with a new system. This is to be chosen when the old system can't continue operation or where off-the-shelf systems would allow the new system to be developed at a reasonable cost.

When you are assessing a legacy system, you have to look at it from both a business perspective and a technical perspective. From a business perspective, you have to decide whether or not the business really needs the system. From a technical perspective, you have to assess the quality of the application software and the system's support software and hardware.

The quality and business value of a system should be assessed according to:

- Low quality, low business value: Keeping the system in operation is expensive and rate of return is small. This system should be scrapped and replaced.
- Low quality, high business value: This system is an important part of the business and cannot be scrapped. It is, however, of low quality and is expensive to maintain. This system should be re-engineered to improve its quality or replaced with a suitable off-the-shelf system.
- High quality, low business value: No huge contribution to the business but is not very expensive to maintain. Normal system maintenance is not expensive and should continue and the system hardware remains unchanged. If changes becomes expensive, the software should be scrapped
- High quality, high business value: This system should be kept in operation. Normal system maintenance should continue.

You would normally choose the replacement option for all or part of a system rather than continue maintenance of the system.

<u>9.6 Explain why problems with support software might mean that an organisation has to replace its legacy systems:</u>

Legacy systems are responsible for the quality software, their application, system's support and hardware. They need to ensure the quality of the support software. Problems with the support software may mean that the functioning or performance of the legacy systems is not as expected. Organisations which depend on many legacy systems and which have a limited budget for maintaining and upgrading these systems have to decide how to get the best return on their investment. This means that they should make a realistic assessment of their legacy systems. So, when problems with support software are found, the organisation thinks of replacing their legacy systems so as to improve their quality.

9.7 As a software project manager in a company that specializes in the development of software for the offshore oil industry, you have been given the task of discovering the factors that affect the maintainability of the systems developed by your company. Suggest how you might set up a program to analyze the maintenance process and discover appropriate maintainability metrics for your company.

Factors which affect maintainability such as (program and data complexity, use of meaningful identifiers, programming language, program documentation etc.). They should then suggest how these can be evaluated in existing systems whose maintenance cost is known and discuss problems of interaction. The approach should be to discover those program units which have particularly high maintenance costs and to evaluate the cost factors for these components and for other components.

The software project manager has to go for at least one offshore industry to which the organisation is going to develop the software. Various issues like the location, environment, people in the industry, users of the software ect are to be noticed personally by going there. By knowing all these issues, the organisation can identify various metrics like time, cost of maintenance, reachability, understandability and usability of the users etc, can be identified. Also some ideas and suggestions from the users are to be known to assess their way usage of the software. By assessing the cost and time to reach to the off-shore industry, the maintenance cost can be assessed.

9.8 Briefly describe the three main types of software maintenance. Why is it sometimes difficult to distinguish between them?

The three main types of software maintenance are:
1. Corrective maintenance or fault repair. The changes made to the system are to repair reported faults which may be program bugs or specification errors or omissions.
2. Adaptive maintenance or environmental adaptation. Changing the software to adapt it to changes in its environment e.g. changes to other software systems.
3. Perfective maintenance or functionality addition. This involves adding new functionality or features to the system.

They are sometimes difficult to distinguish because the same set of changes may cover all three types of maintenance. For example, a reported fault in the system may be repaired by upgrading some other software and then adapting the system to use this new version (corrective + adaptive). The new software may have additional functionality and as part of the adaptive maintenance, new features may be added to take advantage of this.

9.9 Explain the differences between software re-engineering and refactoring

Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code, yet improves its internal structure. It is a disciplined way to clean up code that minimizes the chances of introducing bugs. In essence when you refactor you are improving the design of the code after it has been written." - Martin Fowler (Father of Code Smell).

Although refactoring does not add features or functionalities in a software system, it is sharp weapon for developers in their maintenance activities. It makes a software system easier to understand and cheaper to modify without changing its observable behavior by changing its internal structure.

The purposes of refactoring:

1. Refactoring Improves the Design of Software

2. Refactoring Makes Software Easier to Understand
3. Refactoring Helps Finding Bugs
4. Refactoring Helps Programming Faster

Software Re-engineering means - reorganizing or re-structuring or modifying existing software systems to make them more maintainable.

When to re-engineer:

- When system changes are mostly confined to part of the system then re-engineer that part
- When hardware or software support becomes obsolete
- When tools to support re-structuring are available

The purpose of re-engineering:

1. To explain why software re-engineering is a cost-effective option for system evolution
2. To describe the activities involved in the software re-engineering process
3. To distinguish between software and data re-engineering and to explain the problems of data re-engineering

Re-engineering	Refactoring
It is a maintenance process, so that the understandability and the structure of the program can be improved	The original structure of the program is improved. By doing so, the complexity of the program can be reduced
It can be applied even to legacy software	It is limited to object oriented development programs
You can add any new functionality to the already existing system	Addition of new functionality is not allowed
Reverse engineering is allowed	It is based on agile methods, so reverse engineering is not allowed

9.10 Do software engineers have a professional responsibility to develop code that can be easily maintained even if their employer does not explicitly request it?

Yes, software engineers have a professional responsibility to produce a code that can be maintained and changed. Even if it is not requested by their employer, it is one of their responsibilities to produce such a code. It is their responsibility to develop a code that is easy to understand, maintained, changed and flexible. A code that can be maintained and changed will always be a benefit in future enhancements.

Yes definitely! Software development and maintenance are not separate activities. It is important to keep in mind how a system will need to be maintained if, for example, it is a system that is meant to last a long time and will have a revolving-door of developers working on it. A software developer who cares about the quality of their work will want to reduce the future cost of maintaining their software. According to the textbook, they can do this by using "Good software engineering techniques such as precise specification, test-first development, the use of object-oriented development, and configuration management" to help reduce the cost of future maintenance (Sommerville 259). The ACM Software Engineering Code of Ethics says that a software engineer must "promote an ethical approach to the practice of the profession". This can only be done by carefully constructing software that is in the best interests of the employer (even if they do not require it), the client (who will have to pay for future maintenance), and the public (who want to use the software).

10.1. Suggest six reasons why software dependability is important in most sociotechnical systems.

Six reasons why dependability is important are:

- Users may not use the system if they don't trust it.
- System failure may lead to a loss of business.
- An undependable system may lose or damage valuable data.
- An undependable system may damage its external environment.
- The reputation of the company who produced the system may be damaged hence affecting other systems.
- The system may be in breach of laws on consumer protection and the fitness of goods for purpose.

10.2 Explain with an example why resilience to cyber-attacks is a very important characteristic of system dependability

The dependability of a computer system is a system property that reflects the user's degree of trust in the system. The most important dimensions of dependability are availability, reliability, safety, security, and resilience.

Safe system operation usually depends on the system being available and operating reliably. A system may become unreliable because an intruder has corrupted its data. Denial-of-service attacks on a system are intended to compromise the system's availability. If a system is infected with a virus, you cannot then be confident in its reliability or safety because the virus may change its behavior.

10.3 Using an example, explain why it is important when developing dependable systems to consider these as sociotechnical systems and not simply as technical software and hardware systems.

In a computer system, the software and hardware are interdependent – it is more than the sum of its parts. System dependability is influenced by all of the elements in a sociotechnical system – hardware, software, people and organisations.

The best example to understand the considerations for sociotechnical systems as dependable software is nuclear system used by scientists.

The dependable software is considered as sociotechnical systems because of the following:

- Diversity and redundancy are main components of dependable software. The diversity and redundancy are also main components of sociotechnical systems. So, dependable software is considered as sociotechnical systems
- The repeatable processes are not executed on the basis of individual judgement or interpretation in dependable software. This is also the main ability of sociotechnical systems. The judgement of using repeatable processes is done by team members
- Selection of process model is done in a similar way in both types of systems
- Agile development is used in both systems

10.4 Give two examples of government functions that are supported by complex sociotechnical systems and explain why, in the foreseeable future, these functions cannot be completely automated

Two examples of government functions are Health related services and Department of home affairs. As long as such systems provide services to different types of human users with backgrounds, capabilities, and personalities, these functions cannot be completely automated.

10.5 Explain the difference between redundancy and diversity

Redundancy	Diversity
It means that the spare capabilities of the system can be used if any part of the system is causing failure	It means that there are different types of redundant components in the system, thus increasing the chances that they will not fail in exactly the same way
A similar fault can be repeated	There are different components for the same task, so diversity can't lead to failure

Recovery process is involved	Diversity also involves recovery process but using different components
Redundant components are involved in software systems to ensure the same functionality with respect to other components of the system	Different component with the same functionality are involved in diversity

10.6 Explain why it is reasonable to assume that the use of dependable processes will lead to the creation of dependable software

Dependable software processes ensure the critical systems have been properly enacted, documented, and developed using appropriate techniques

Dependable software processes are software processes that are designed to produce dependable software. Dependable software processes often include different activities that have the same aim. The dependability of the final software is directly proportional to the dependability of the individual processes. As each of the software processes used are dependable, the software produced will also be dependable software

10.7 Give two examples of diverse, redundant activities that might be incorporated into dependable processes.

Agile based development using non-object-oriented programming

Plan driven development using object-oriented programming

10.8 Give two reasons why different versions of a system based on software diversity may fail in a similar way

1. The system may include explicit diversity policies that are intended to maximize the differences between the system versions. The diverse

versions of the system should have no dependencies and so should fail in completely different ways.
In this case overall reliability of a diverse system is obtained by multiplying the reliabilities of each channel. If each channel has a probability of failure on demand, then the overall POFOD of the 3-channel is a million time greater than the reliability of a single channel system

2. Achieving the complete channel independence is impossible. It has been shown experimentally that independent design team often make the mistakes or misunderstands the same of the specification
    - If the requirements are incorrect or they are based on misunderstanding about the environment of the system, then these mistakes will be reflected
    - In a critical system, the V-space is a detailed document based on the system's requirements, which provides full details to the teams on how the system should behave.
    - There is no scope for interrelation by the software developers, if there are errors in this document, and then these will be presented to all of the development teams and implemented in all versions of the system

10.9 You are an engineer in charge of the development of a small, safety-critical train control system, which must be demonstrably safe and secure. You suggest that formal methods should be used in the development of this system, but your managers is skeptical of this approach. Write a report highlighting the benefits of formal methods and presenting a case for their use in this project.

Formal methods are mathematical approaches to software development where you define a formal method of the software. You may then formally analyze this model to search for errors and inconsistencies, prove that a program is consistent with this model, or you may apply a series of correctness-preserving transformations to the model to generate a program.

The advantages of developing a formal specification and using it in a formal development process are:

1. As you develop a formal specification in detail, you develop a deep and detailed understanding of the system requirements. Requirements problems that are discovered early are usually much cheaper to correct than if they are found at later stages in the development process
2. As the specification is expressed in a language with formally defined semantics, you can analyze it automatically to discover inconsistencies and incompleteness
3. If you use a method such as the B method, you can transform the formal specification into a program through a sequence of correctness-preserving transformations. The resulting program is therefore guaranteed to meet its specification
4. Program testing costs may be reduced because you have verified the program against its specification. For example, in the development of the software for the Paris Metro systems, the use of refinement meant that there was no need for software component testing and only system testing was required.

During the use of formal methods fewer errors in the delivered software were reported.

This system can be used in most of the super-fast train systems to ensure their safety. As it uses regular language, so, it can be easily understood by the train operators.

10.10 It has been suggested that the need for regulation inhibits innovation and that regulators force the use of older methods of systems development that have been used on other systems. Discuss whether or not you think this is true and the desirability of regulators imposing their views on what methods should be used.

In some cases it is true that regulators force the use of already used or older methods for system development. The main reason for this is that regulators are mostly used in security and safety systems. If the method used to develop a new system is already implemented in any other system, then all the drawback of this

system can be understood in an effective manner. The new system will be free from all such errors which are faced in the past. If any system is successfully implemented in the past, then there is no worry to use the features of that system in newly developed systems.

Depending on the requirements of the system, sometimes, regulators impose the view of any system by defining the method used for development. In most of the cases, regulators force use formal method for development. The reason behind this is that the system requirements for the system can be fully understood in a detailed and deep manner

Regulators use cost effective methods for testing the system. By using formal methods, the testing cost will be minimized. This is because, the program is already verified against its specifications.

Chapter 11

11.1 Explain why it is practically impossible to validate reliability specifications when these are expressed in terms of a very small number of failures over the total lifetime of a system

To measure reliability you need to have statistically valid failure data for the system so you need to induce more failures than are specified in the given time period. However, because the number of failures is so low, this will take an unrealistically large amount of time.

11.2 Suggest appropriate reliability metrics for the classes of software system below. Give reasons for your choice of metric. Predict the usage of these systems and suggest appropriate values for the reliability metrics.

- a system that monitors patients in a hospital intensive care unit
- a word processor
- an automated vending machine control system
- a system to control braking in a car
- a system to control a refrigeration unit
- a management report generator

System	Reliability metric	Suggested value	Rationale
Patient monitoring system	Availability	System should be unavailable for less than 20 minutes per month.	The system needs to be continuously available as patients may be admitted or discharged at any time. The chosen figure is acceptable because, if necessary, critical system functions can be taken over manually.
Word processor	ROCOF	Failures resulting in loss of data should not occur more than once per 1200 hours of use.	
Vending machine controller	POFOD (Probability of failure on demand)	Failure acceptable in 1:5000 demands	Not a critical system so relatively high failure rate is OK.
Braking system controller	POFOD	The software should never fail within the predicted lifetime of the system.	Very critical system. Failure is unacceptable at any time.
Refrigeration unit control	Availability	20 minutes per month	Non-stop system but not critical. Short periods of failure are not a real problem as temperature takes some time to rise.
Management report generator	ROCOF	1 fault/100 hours of use	Not a critical system. Faults are unlikely to cause severe disruption.

11.3 Imagine that a network operations center monitors and controls the national telecommunications network of a country. This includes controlling and monitoring the operational status of switching and transmission equipment and keeping track of nationwide equipment inventories. The center needs to have redundant systems. Explain three reliability metrics you would use to specify the needs of such a system

1. Probability of failure on demand (POFOD):
   If you use this metric you define the probability that a demand for service from a system will result in a system failure. So, POFOD = 0.001 means that there is a 1/1000 chance that a failure will occur when a demand is made

2. Rate of occurrence of failures (ROCOF):
   This metric sets out the probable number of system failures that are likely to be observed relative to a certain time period, or to the number of system executions. The reciprocal of ROCOF is the mean time to failure (MTTF), which is sometimes used as a reliability metric. MTTF is the average

number of time units between observed system failures. A ROCOF of two failures per hour implies that the mean time to failure is 30 minutes

3. Availability (AVAIL):
   Availability is the probability that a system will be operational when a demand is made for service. Therefore, an availability of 0.999 means that, on average, the system will be available for 99.99% of the operating time.

11.4 What is the common characteristic of all architectural styles that are geared to supporting software fault tolerance?

The common characteristics of all styles to support fault tolerance is that there are multiple separate implementations of system functionality and some error detection mechanism that can detect possible software failures

In general, fault-tolerant software by implanting fault tolerance techniques share the following characteristics:

1. Many details of their implementation are made transparent to the users
2. They provide well-defined interfaces for the definition and implementation of fault tolerance techniques
3. They are recursive in nature

11.5 Suggest circumstances where it is appropriate to use a fault-tolerant architecture when implementing a software-based control system and explain why this approach is required.

Fault: this is the characteristic of a software system whose occurrence can result in a system error
Fault tolerance: is a runtime approach to dependability in which systems include mechanisms to continue in operation, even after a software or hardware fault has occurred and the system state is erroneous. Fault tolerance mechanisms detect and correct this erroneous state so that the occurrence of a fault does not lead to a system failure

The fault tolerance system architecture are used whenever there is a requirement of high level system availability and reliability for example Aircraft Flight control system

In aircraft flight control system computations are carried out in parallel by each of the flight control computers while using the same inputs. There are hardware filters connected to the outputs which can detect if a fault condition occurs. If a fault in one computer occurs, output can be taken from other alternative system without disturbing the operation

11.6 You are responsible for the design of a communications switch that has to provide 24/7 availability but that is not safety-critical. Giving reasons for your answer, suggest an architectural style that might be used for this system

The most appropriate architectural pattern is an N-version programming architecture or a replicated server architecture with each server running a different OS.

A self-monitoring architectural style can be used. A self-monitoring architecture is a system architecture in which the system is designed to monitor its own operation and to take some action if a problem is detected.
For a system which require availability this architecture is more suitable, because even if some failure occur the system is designed in a way to resist and recover. As the system is not a safety critical one, protection system architecture does not suit well

11.7 It has been suggested that the control software for a radiation therapy machine, used to treat patients with cancer, should be implemented using N-version programming. Comment on whether or not you think this is a good suggestion

Advantages of N-version programming

1. Increases design diversity so probability of faults that result in failures should be reduced

2. Increases availability of the system

Disadvantages

1. Increased cost because of the need to use independent development teams
2. Increased software complexity because of the need for a fault tolerant controller. Increased complexity increases the probability of error
3. Improvement in reliability in practice is limited because of the possibility of common errors made by different development teams.

N-version programming would not be a good design strategy for this type of software. There is no need for high availability and the increased complexity and cost would make the overall cost of the machine too high.

11.8 Explain why all the versions in a system designed around software diversity may fail in a similar way

- There may be a specification error that is reflected in both versions.
- The problem may be a numeric error that has not been explicitly trapped.
- The specification may be ambiguous and may be misunderstood in the same way by both teams.

11.9 Explain how programming language support of exception handling can contribute to the reliability of software systems

An error or an unexpected event that occurs during the execution of a program is called an exception. Exceptions may be caused by hardware or software conditions. When an exception occurs, it must be managed by the system.

Language such as Java, C++ and Python have built-in exception-handling constructs. When an exceptional situation occurs, the exception is signaled and the language runtime system transfers control to an exception handler. This is a code section that states exception names and appropriate actions to handle each

exception. The exception handler is outside the normal flow of control, and this normal control flow does not resume after the exception has been handled.

Handling exception within a program makes it possible to detect and recover from some input errors and unexpected external events. As such, it provides a degree of fault tolerance. The program detects faults and can take action to recover from then. As most input errors and unexpected external events are usually transient, it is often possible to continue normal operation after the exception has been processed.

11.10 Software failures can cause considerable inconvenience to users of the software. Is it ethical for companies to release software that they know includes faults that could lead to software failures? Should they be liable for compensating users for losses that are caused by the failure of their software? Should they be required by law to offer software warranties in the same way that consumer goods manufacturers must guarantee their products?

Ethics for software failure:
Failure is a state of the system due to which a system stops functioning and it was raised by a fault in software application. If any organisation is releasing any software with faults and they knew it, it is highly unethical:

- It can lead to great financial loses to the customer
- The trust of the customer towards the company will not be maintained
- Competitors can take advantage of this
- It can lead to ethical and financial loses to the developing company

Yes, the company is liable for the user's loss if the company is doing this knowingly. The customer is not aware of problems associated with the software, so the user will not bother for this type of incidents. The company should not have released any software product if the product as bugs which are leading system failures.

Yes it is possible that companies can provide warranties for their software product like other products. But the beta version will never contain any warranty.

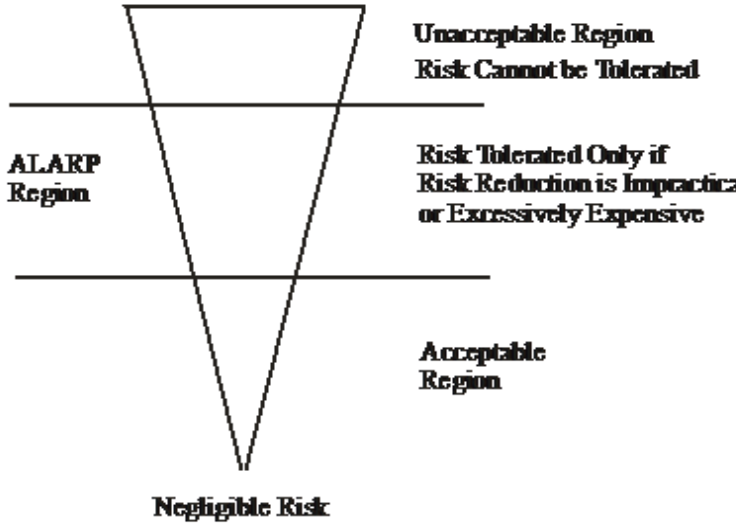
Chapter 12

12.1 Identify six consumer products that are likely to be controlled by safety-critical software systems

Microwave oven
Power tools such as a drill or electric saw
Lawnmower
Central heating furnace
Garbage disposal unit
Food processor or blender

12.2 A software system is to be deployed for a company that has extremely high safety standards and allows for almost no risks, not even minor injuries. How will this affect the look of the risk triangle in figure 12.3?

Risk triangle:



The boundaries between the regions are not fixed but depend on how acceptable risks are in the societies where the system will be deployed. The boundaries in the risk triangle are liable to change with time and changing social attributes, due to the acceptance or negligence of a risk sometimes depend on the time and the risk that occurred. At some point in time the risk may appear as tolerable but sometimes the severity may be high. The acceptance and assessment depends on the changing minds of the people.

Over time, society has become more risk averse so the boundaries have moved downward. Although the financial costs of accepting risks and paying for any resulting accidents may be less than the costs of accident prevention, public opinion may demand that money be spent to reduce the likelihood of a system accident, thus incurring additional costs. For example, it may be cheaper for accompany to clean up pollution on the rare occasion it occurs, rather than to install systems for pollution prevention

12.3 In the insulin pump system, the user has to change the needle and insulin supply at regular intervals and may also change the maximum single dose and the maximum daily dose that may be administered. Suggest three user errors that might occur and propose safety requirements that would avoid these errors resulting in an accident

Possible user errors are:

1. Maximum daily dose set wrongly
2. Maximum single dose set wrongly
3. Failure to replace empty insulin reservoir
4. Insulin reservoir improperly fitted
5. Needle improperly fitted

Examples of safety requirements to avoid these errors are:

1. When the maximum dose and the maximum daily dose is changed, the user should be asked to input the changed values twice.

2. If the maximum daily dose has already been set by the user then the new daily dose should be no more than 1.25 and no less than 0.75 of the previous maximum daily dose.
3. The insulin reservoir case should be designed so that it is only possible to fit the insulin bottle the right way and the case should not close unless the bottle is properly seated.
4. If the back pressure from the needle assembly is more than XX then the system should shut down and issue an audible and text warning. This allows for blocked needles as well as improperly fitted needles.

12.4 A safety-critical software system for managing roller coasters controls two main components:

- The lock and release of the roller coaster harness which is supposed to keep riders in place as the coaster performs sharp and sudden moves. The roller coaster could not move with any unlocked harnesses.
- The minimum and maximum speeds of the roller coaster as it moves along the various segments of the ride to prevent derailing, given the number of people riding the roller coaster.

Identify three hazards that may arise in this system. For each hazard, suggest a defensive requirement that will reduce the probability that these hazards will result in an accident. Explain why your suggested defense is likely to reduce the risk associated with the hazard.

Hazard 1: Lock and release of the harness becomes unlocked during a ride

Continuous use of a sensor to observe when a harness become unlocked and bring the ride to a slower movement until a stop. Ensure that the roller coaster does not come to a sudden stop but reduce speed first and then come to a stop on a safe point in the roller coaster track

Hazard 2: Number of people is mistakenly calculated

Weight sensors added to the roller coaster to accurately measure the actual weight being carried by the roller coaster to ensure an Arithmetic error is avoided

Hazard 3:

12.5. A train protection system automatically applies the brakes of a train if the speed limit for a segment of track is exceeded, or if the train enters a track segment that is currently signaled with a red light (i.e., the segment should not be entered). There are two critical-safety requirements for this train protection system:

- The train shall not enter a segment of track that is signaled with a red light.
- The train shall not exceed the specified speed limit for a section of track.

Assuming that the signal status and the speed limit for the track segment are transmitted to on-board software on the train before it enters the track segment, propose five possible functional system requirements for the onboard software that may be generated from the system safety requirements.

The five possible functional system requirements for the onboard software that may be generated from the system safety requirements:

1. Insurance by the system that if a red signal is received, train brakes are applied
2. Whenever a red signal is received there should be an alarm sound in the driver's cabin
3. The train speed should be compared with segment speed limit once per second by the system
4. The train brakes should be applied if the speed of the train exceeds the segment speed limit
5. The train brakes should be applied if the train deceleration is less than comfortable deceleration

12.6 Explain when it may be cost-effective to use formal specification and verification in the development of safety-critical software systems. Why do you think that some critical systems engineers are against the use of formal methods?

Formal methods can be cost-effective in the development of safety-critical software systems because the costs of system failure are very high and so additional cost in the development process is justified. Most safety-critical systems have to gain regulatory approval before they are used and it is a very expensive process to convince a regulator that a system is safe. The use of a formal specification and associated correctness argument may be less than the costs e.g. of additional testing to convince the regulator of the safety of the system.

Some developers of systems are against the use of formal methods because they are unfamiliar with the technology and unconvinced that a formal specification can be complete representation of the system. Furthermore, the problem with formal specifications are that they cannot be understood by system customers so they may conceal errors and give a false picture of the correctness of the system.

12.7 Explain why using model checking is sometimes a more cost-effective approach to verification than verifying a program's correctness against a formal specification.

Formally verifying programs using a deductive approach is difficult and expensive but alternative approaches to formal analysis have been developed that are based on a more restricted notion of correctness.

Model checking has been widely used to check hardware systems designs. It is increasingly being used in critical software systems such as the control software n Nasa's Mars exploration vehicles.
If the model has to be created manually, it is an expensive process as model creation take a great deal of time. It is therefore best if the model can be create automatically from the program source code. Model checkers are available that work directly from programs in Java, C, C++ and Ada.

As more algorithms are incorporated into model checkers, it will be increasingly possible to use model checking routinely in large-scale systems development

12.8. List four types of systems that may require software safety cases, explaining why safety cases are required

Many safety-critical, software-intensive systems are regulated. An external authority has significant influence on their development and deployment. Regulators are government bodies whose job is to ensure that commercial companies do not deploy systems that pose threats to public and environmental safety or the national economy. The owners of safety-critical systems must convince regulators that they have made the best possible efforts to ensure that their systems are safe. The regulator assesses the safety case for the system, which presents evidence and arguments that normal operation of the system will not cause harm to a user.

A safety case is a set of structured documents that includes a description of the system to be certified. It contains the information about the processes used to develop the system and logical arguments that demonstrate that the system is safe or that a required level of security is achieved. Therefore for many critical systems the safety case is a legal requirement. Software failures can result in failures of equipment or other processes which may lead to injury or death. Safety cases may vary depending on the type of system.

The systems that may require safety cases are:

- Banking system
- Systems used to control equipment in the nuclear industry where there is a possibility of the release of radioactivity
- Health care system
- Air-traffic control systems
- Signaling and control systems in the railway industry
- Software for critical aircraft functions such as flight control systems.

12.9. The door lock control mechanism in a nuclear waste storage facility is designed for safe operation. It ensures that entry to the storeroom is only permitted when radiation shields are in place or when the radiation level in the room falls below some given value (dangerLevel).

So:
(i) If remotely controlled radiation shields are in place within a room, an authorized operator may open the door.
(ii) If the radiation level in a room is below a specified value, an authorized operator may open the door.
(iii) An authorized operator is identified by the input of an authorized door entry code.

The code shown in Figure 12.15 controls the door-locking mechanism. Note that the safe state is that entry should not be permitted. Using the approach discussed in this chapter, develop a safety argument for this code. Use the line numbers to refer to specific statements. If you find that the code is unsafe, suggest how it should be modified to make it safe.

```
1       entryCode = lock.getEntryCode () ;
2       if (entryCode == lock.authorizedCode)
3       {
4               shieldStatus = Shield.getStatus ();
5               radiationLevel = RadSensor.get ();
6               if (radiationLevel < dangerLevel)
7                       state = safe;
8               else
9                       state = unsafe;
10              if (shieldStatus == Shield.inPlace() )
11                      state = safe;
12              if (state == safe)
13                      {
14                              Door.locked = false ;
15                              Door.unlock ();
16                      }
17              else
18              {
19                      Door.lock ( );
20                      Door.locked := true ;
21              }
22      }
```

There are two potential safety problems with this code:

1 Say the door was unlocked when the door entry code was entered.

Line 13 checks if the state is safe and, if it is safe then unlocks the door. However, if the door was unlocked to begin with, there is no locking action if the state is unsafe so therefore a potential safety loop hole exists.

2 If the radiation level is less than the danger level then line 8 sets the state to be safe. However, line 10 checks the shields to see if they are in place. If they are not in place, the state is unchanged although, in fact, the system is unsafe if the shields are down. Therefore, the door can be opened with the shields down and a safety loophole exists.

There are two changes which should be made to ensure that the code is safe:

- An initial statement which locks the door and sets door locked to be true.
- The if statement if shield-status == Shield.inPlace then should be changed to:

```
if (shield_status == Shield.inPlace())
  state := safe;
else
  state := unsafe;
```

12.10. Should software engineers working on the specification and development of safety-related systems be professionally certified or licensed in some way? Explain your reasoning.

Just about every electrical device today contains software. A flat-screen television has thousands of lines of code and more than 10 processors. A mid-priced car can have as many as 20. Medical equipment, elevators, and even microwave ovens are run by software. But a poorly written program or a software glitch can cause a lot of harm when a car's braking system fails, for example, or an insulin pump stops working, or an elevator gets stuck between floors.

If software engineers who write programs for systems that expose the public to physical or financial risk knew they would be tested on their competence, the thinking goes, it would reduce the flaws and failures in code—and maybe save a few lives in the bargain. That's why IEEE has been working on an initiative to license software engineers who pass a competency exam.

There are several ways to view the differences between licensure and certification, but there are a couple of aspects that, for me, are most significant. First, there is the legal aspect: licensed engineers are, in most circumstances, the only ones who have the authority to take legal responsibility for engineering work. Licensure is sanctioned by government agencies as opposed to certification, which is administered by private organizations. A licensed engineer takes personal responsibility for submitted engineering work and, of course, the liability that goes with that.

The other aspect is related to ethics. Licensed engineers are bound to a set of ethical standards that are enforced by state licensing boards under the threat of loss of licensure or legal action.

Both of these aspects are unique to licensure and I believe they drive the behavior and decisions made by the licensed engineer profoundly. This means that a client has a dimension of assurance and transparency when they seek engineering judgment from a licensed engineer about questions of safety and risk. Legally and ethically, licensed engineers are bound to represent to clients their full assessment, despite corporate spin or economic pressures.

If one acknowledges that the application and use of software play a significant role in system safety, then it seems reasonable to apply licensure regulation to software engineering within the same framework that is applied to other engineering disciplines.

Chapter 15

15.1. What major technical and nontechnical factors hinder software reuse? Do you personally reuse much software and, if not, why not?

Technical issues tend to focus on how reusable the software is. This means the potential of software for adaptation. Non-technical issues include how a project is managed and funded.

The non-technical problems in reusing components include increased maintenance costs, finding understanding, and adapting reusable components. If the source code of a reused software system is not available, then maintenance costs may be higher
Technical issue is not the one for reusability to reach its full potential, non-technical factors are also to be considered. These factors are related to the motivational and organizational issues. Managers may be unwilling to compromise their requirements to allow reusable components to be used. Also there may be lack of tool support

Software reuse must be considered when development cost and time can be reduced. But there is also a significant cost associated with whether or not a components is suitable for reuse in a particular situation. In that situation software reuse must not be considered.

15.2. List the benefits of software reuse and explain why the expected lifetime of the software should be considered when planning reuse.

Benefit	Explanation
Accelerated development	Bringing a system to market as early as possible is often more important than overall development costs. Reusing software can speed up system production because both development and validation time may be reduced.
Effective use of specialists	Instead of doing the same work over and over again, application specialists can develop reusable software that encapsulates their knowledge.
Increased dependability	Reused software, which has been tried and tested in working systems, should be more dependable than new

	software. Its design and implementation faults should have been found and fixed.
Lower development costs	Development costs are proportional to the size of the software being developed. Reusing software means that fewer lines of code have to be written.
Reduced process risk	The cost of existing software is already known, while the costs of development are always a matter of judgment. This is an important factor for project management because it reduces the margin of error in project cost estimation. This is especially true when large software components such as subsystems are reused.
Standards compliance	Some standards, such as user interface standards, can be implemented as a set of reusable components. For example, if menus in a user interface are implemented using reusable components, all applications present the same menu formats to users. The use of standard user interfaces improves dependability because users make fewer mistakes when presented with a familiar interface.

15.3. How does the base application's design in the product line simplify reuse and reconfiguration?

Base applications are often developed with some objectives in mind and one of the objectives to be kept in mind is the simplicity of the design of the application because it is not one man who is involved in the process of the development and creation and coding of the application, there are teams of individuals who are

assigned a specific task and with change in the team if the design becomes complicated then there is a risk that the end product will be so very complicated that it practically impossible for any person to understand and deploy it therefore the first and primary objective of the design or development or deployment or testing team is to keep the base and emerging application to simple so that if there is any change in the existing team is there then it becomes simpler for the new incoming team to understand the past work and continue the present work in hand and deliver the product in/before time to the customer. Re-use of the existing code/components/applications-application families or other families, objects and functions leads to benefits of increase in dependability, increase in process risk, use of specialists in effective manner. In the product line the simplification in the base design benefits reuse and reconfiguration in the way:

- components can be reused in other applications dependent on base application as the components will be existing in the base library and they just have to be inherited and used.
- Functions that are declared as global in the base application can be reused just by simple call function procedures.
- since the components and functions are already existing therefore populating the re-usable component, function, application library is less time consuming and the development time is accelerated.
- for reconfiguration purposes changes in the hardware is less required or might not be required at all.
- Documentation needs not to be written at all.
- increase in the life time expectancy of the software.

15.4. Explain what is meant by “inversion of control” in application frameworks. Explain why this approach could cause problems if you integrated two separate systems that were originally created using the same application framework.

To implement a system using a framework, you add concrete classes that inherit operations from abstract classes in the framework. In addition, you define “callbacks”— methods that are called in response to events recognized by the framework. The framework objects, rather than the application-specific objects, are responsible for control in the system. This is called “inversion of control.”

The inversion of control occurs when two or more frameworks call the application code simultaneously, each assuming ownership of the application's main event loop.

15.5. Using the example of the weather station system described in Chapters 1 and 7, suggest a product-line architecture for a family of applications that are concerned with remote monitoring and data collection. You should present your architecture as a layered model, showing the components that might be included at each level.

#	Layers	Clarification
Layer 0	**I/O management:** Sensor data input	Report generation
Layer 1	**Data management:** Data collection, data monitoring, transaction control	Controlling the collection and monitoring of the data.
Layer 2	**Instruments** Drivers for each of the tools	Managing the specific tools in the system.
Layer 3	**Data storage:** Data storage, data compression, data protection, data integrity provision	Controlling the data that has been collected.
Layer 4	**Interactions:** Operating interfaces	Remote system interaction.

1. Communications (the lowest layer). All components that are concerned with interaction with a remote system.
2. Data storage and management. All components that are concerned with looking after the data that has been collected.
3. Instruments: all components that are concerned with managing the specific instruments in the system
4. Data collection and processing. All components that control the collection, processing and monitoring of the collected data

Data collection and processing

Initializing Data Collection Data monitoring

Instruments

Specific divers for each of the instruments In the system

Data storage and management

Data	Data storage

Data Compression Validation

Communications

Upload data	Download
Remote	Commands
Management	

15.6. Most desktop software, such as word processing software, can be configured in a number of different ways. Examine software that you regularly use and list the configuration options for that software. Suggest difficulties that users might have in configuring the software. Microsoft Office (or one of its open-source alternatives) is a good example to use for this exercise.

Problem	Clarification
Typing, Spellchecking and Grammar Checking problems	Typing, spellchecking and grammar checking in any other language, but English, French and Spanish.
Inserting a symbol causes Word to crash sometimes	Word crashes with the error message “Winword has encountered an error and needs to close. Error details: hpxxxx.dll or hpxxxx.drv”
Fields Don’t Display Correctly after Mail merge	When performing mailmerge in Word with a database from Excel or Access, some of the fields in Word will not display correctly.
Applying Bullets and Numbering causes Word to crash	Clicking on bullets and numbering on the Formatting toolbar in Word causes application to crash or exhibits unusual behavior for bullets and numbering.
Word files open/save slowly	Word files open slowly. Retrieving or saving a Word file takes unusually long time.
Unexpected/unusual behavior in Word	In Word, File, Edit, or View menus may open and close without keyboard or mouse input from users. Text spontaneously appears in Word without any input from users.
Errors on opening Clipart/Clip Organizer	Clicking on Clipart or Clip Organizer in Word, causes the error message “Clip art organizer cannot complete the requested task” or “Clip Organizer cannot complete the operation” to appear.
Problems with access privileges	Customer wants to create a document, where some portions of the document have to be filled in by users, whereas some portions of text would be restricted to be accessed.

Macros in this project are disabled	Error message “The macros in this project are disabled. Please refer to the online help or documentation of the host application to determine how to enable macros.” upon launching or exiting Word.
Toolbars and Menus are Missing	Toolbars and menus are missing upon launching Word.
Fail to open a doc file	Word document will not open. Word document appears to be corrupted, unreadable or damaged.

15.7. Why have many large companies chosen ERP systems as the basis for their organizational information system? What problems may arise when deploying a large-scale ERP system in an organization?

Many large companies chosen ERP systems as the basis for their organizational information system because Enterprise Resource Planning (ERP) system may support all of the manufacturing, ordering, and customer relationship management activities in a large company. With the help of ERP all the resources companies possess could be allocated in much more effective and profitable way. It is all about getting more benefits from the available resources. No wonder large companies interested in using it, because the only aim they pursue is making more money.

A problem that may arise when deploying a large-scale ERP system in an organization is that ERP systems usually require extensive configuration to adapt them to the requirements of each organization where they are installed.

15.8. What are the significant benefits offered by the application system reuse approach when compared with the custom software development approach?

Benefit	Description
Accelerated development	Bringing a system to market as early as possible is often more important than overall development costs. Reusing software can speed up system production because both development and validation time may be reduced.
Effective use of specialists	Instead of doing the same work over and over again, application specialists can develop reusable software that encapsulates their knowledge.
Increased dependability	Reused software, which has been tried and tested in working systems, should be more dependable than new software. Its design and implementation faults should have been found and fixed
Lower development costs	Development costs are proportional to the size of the software being developed. Reusing software means that fewer lines of code have to be written.
Reduced process risk	The cost of existing software is already known, while the costs of development are always a matter of judgment. This is an important factor for project management because it reduces the margin of error in project cost estimation. This is especially true when large software components such as subsystems are reused.
Standards compliance	Some standards, such as user interface standards, can be implemented as a set of reusable components. For example, if menus in a user interface are implemented using reusable components, all applications present the same menu formats to users. The use of standard user interfaces improves dependability because users make fewer mistakes when presented with a familiar interface.

This approach to software reuse has been very widely adopted by large companies as it offers significant benefits over customized software development:

- More rapid deployment of a reliable system may be possible
- It is possible to see what functionality is provided by the applications, and so it is easier to judge whether or not they are likely to be suitable. Other companies may already use the applications, so experience of the system is available.

- Some development risks are avoided by using existing software.
- Businesses can focus on their core activity without having to devote a lot of resources to IT systems development
- As operating platforms evolve, technology updates may be simplified as these are the responsibility of the application system vendor rather than the customer.

15.9. Explain why adaptors are usually needed when systems are constructed by integrating application
Systems. Suggest three practical problems that might arise in writing adaptor software to link two application systems.

When more than two application systems are combined for performing the requirements of the user, then that system is called an integrated system. To make integrated systems, adaptors are required.

- Adaptors are used to combine the newly developed system with already existing systems
- Adaptors acts as an API between new system and existing system.
- For interaction among system components for both the systems, adaptors are required.
- In some systems, the output of existing system is required as input to newly developed system, so, adaptors are required.

Different systems normally use unique data structures and formats. You have to write adaptors that convert from one representation to another. These adaptors are runtime systems that operate alongside the constituent application systems.

Application system integration problems:

- Lack of control over functionality and performance
  - Application systems may be less effective than they appear
- Problems with application system inter-operability
  - Different application systems may make different assumptions that means integration is difficult
- No control over system evolution
  - Application system vendors not system users control evolution
- Support from system vendors

- Application system vendors may not offer support over the lifetime of the product

15.10. The reuse of software raises a number of copyright and intellectual property issues. If a customer pays a software contractor to develop a system, who has the right to reuse the developed code?
Does the software contractor have the right to use that code as a basis for a generic component?
What payment mechanisms might be used to reimburse providers of reusable components?
Discuss these issues and other ethical issues associated with the reuse of software.

This would be different from situation to situation. The original specs of the first project and the contract between the parties involved would determine the future of the software. The contractor I believe should have a right to use the generic parts of the system, but anything specific or specialized for the client would be the property of the client. If there is a profit to be made on future versions, royalties would be a basic way of compensation for reusable components.

It is very easy to cross the line from ethical actions to unethical actions by reusing software. Software that is generic(core components) can easily be reused, but once you have created a functioning product and have working code that belongs to another client, the temptation might be there to use what you have already done. There is also the possibility that the contractor could use the specialized application components accidentally. By already creating it previously, the contractor might just subconsciously use similar or exact methods that he/she might now own the rights to.

Owning the rights to component design (and associated algorithms) is illogical. The software contractor could potentially reuse a purchased component and it is unlikely that the customer would ever know.

Chapter 18

18.1. Why is it important to define exceptions in service engineering?

Abstract interface design starts with the service requirements and defines the operation names and parameters. At this stage, you should also define the exceptions that may arise when a service operation is invoked. Defining exceptions and how these exceptions can be communicated to service users is particularly important. Service engineers do not know how their services will be used. It is usually unwise to make assumptions that service users will have completely understood the service specification. Input messages may be incorrect, so you should define exceptions that report incorrect inputs to the

service client. It is generally good practice in reusable component development to leave all exception handling to the user of the component. Service developers should not impose their views on how exceptions should be handled.

18.2. Standards are fundamental to service-oriented architectures, and it was believed that standards conformance was essential for successful adoption of a service-based approach. However,
RESTful services, which are increasingly widely used, are not standards-based. Discuss why you think this change has occurred and whether or not you think that the lack of standards will inhibit the development and takeup of RESTful services.

The term Rest arises for Representational State Transfer. The Restful services are not based on standards. The points to show the reasons for using Restful services instead of service-based approach are as follows:

1. The standards to develop software are changing on a regular bases. In RESTful services each and everything represents a resource. This the main reason behind this change
2. A unique identifier is used in RESTful services architecture. The WWW is the best example of this
3. The data is directly accessed in RESTful services in place of making any action for any particular value of data
4. RESTful services are not XML based.

No, the RESTful services will not be inhibited because of lack-of-standards. It has specific standards of its own. In today's world, most of the applications are web based. The RESTful services provide URL queries to directly access data, which makes it fast and easy to use. So, it can be said that RESTful services are never reversed or inhibited.

18.3. Extend Figure 18.5 to include WSDL definitions for MaxMinType and InDataFault. The temperatures should be represented as integers, with an additional field indicating whether the temperature is in degrees Fahrenheit or degrees Celsius. InDataFault should be a simple type consisting of an error code.

WSDL for InDataFault and MaxMinType: the WSDL stands for Web Service Description Language. The extended WSDL definions for given figure are:

```
<xs: complexType name = “MaxMinType”
        <xs: sequence>
                <xs:element name = “maxtemp” type = “xs:integer”/>
                <xs:element name = “mintemp” type = “xs:integer”/>
<!--  Assume that an enumerated type called tempscale with values Celsius and
        <xs: complexType name = “InDataFault”
         <xs: sequence>
                <xs:element name = “errorcode” type = “xs:integer”/>
```

8.4. Suggest how the SimpleInterestCalculator service could be implemented as a RESTful service.

18.5. What is a workflow? List out the key stages in the process of system construction by composition.

You can think of this process as a sequence of separate steps. Information is passed from one step to the next. For example, the rental car company is informed of the time that the flight is scheduled to arrive. The sequence of steps is called a workflow—a set of activities ordered in time, with each activity carrying

out some part of the work. A workflow is a model of a business process; that is, it sets out the steps involved in reaching a particular goal that is important for a business. In this case, the business process is the vacation booking service, offered by the airline.

The six key stages in the process of system construction by composition:
1. *Formulate outline workflow:* In this initial stage of service design, you use the requirements for the composite service as a basis for creating an "ideal" service design. You should create a fairly abstract design at this stage, with the intention of adding details once you know more about available services.

2. *Discover services:* During this stage of the process, you look for existing services to include in the composition. Most service reuse is within enterprises, so this may involve searching local service catalogs. Alternatively, you may search the services offered by trusted service providers, such as Oracle and Microsoft.

3. Select possible services: From the set of possible service candidates that you have discovered, you then select possible services that can implement workflow activities. Your selection criteria will obviously include the functionality of the services offered. They may also include the cost of the services and the quality of service (responsiveness, availability, etc.) offered.

4. *Refine workflow:* On the basis of information about the services that you have selected, you then refine the workflow. This involves adding detail to the abstract description and perhaps adding or removing workflow activities. You may then repeat the service discovery and selection stages. Once a stable set of services has been chosen and the final workflow design established, you move on to the next stage in the process.

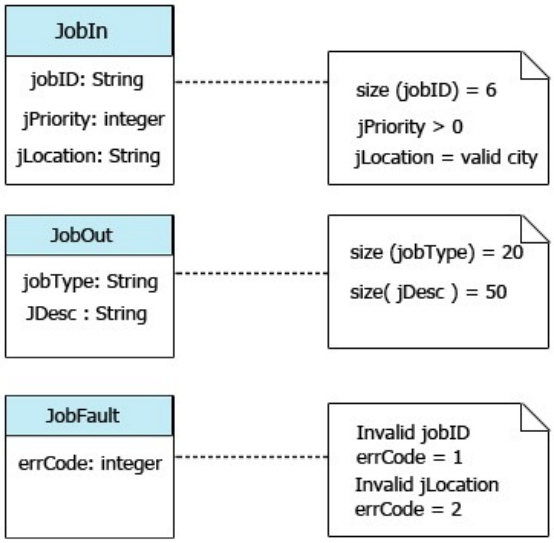
5. *Create workflow program:* During this stage, the abstract workflow design is transformed to an executable program and the service interface is defined. You can implement workflow programs using a programming language, such as Java or C#, or by using a workflow language, such as BPMN (explained below). This stage may also involve the creation of web-based user interfaces to allow the new service to be accessed from a web browser.

6. *Test completed service or application:* The process of testing the completed, composite service is more complex than component testing in situations where external services are used.

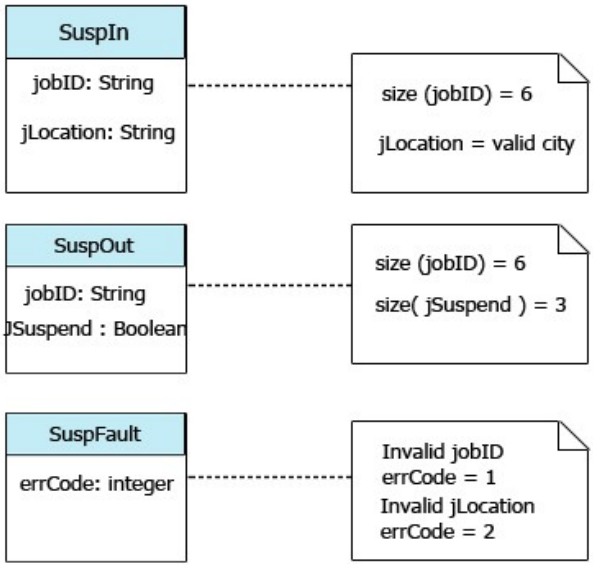
18.6. Design possible input and output messages for the services shown in Figure 18.13. You may specify these in the UML or in XML.

Input and output messages for Maintenance service in UML notation

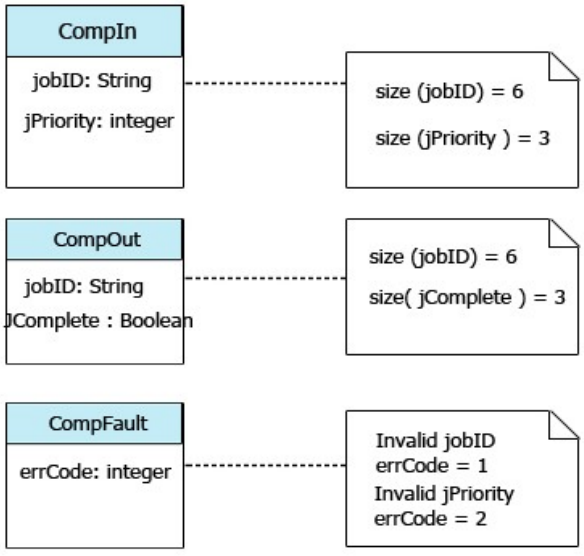
1. getJob Operation



2) SuspendJob operation

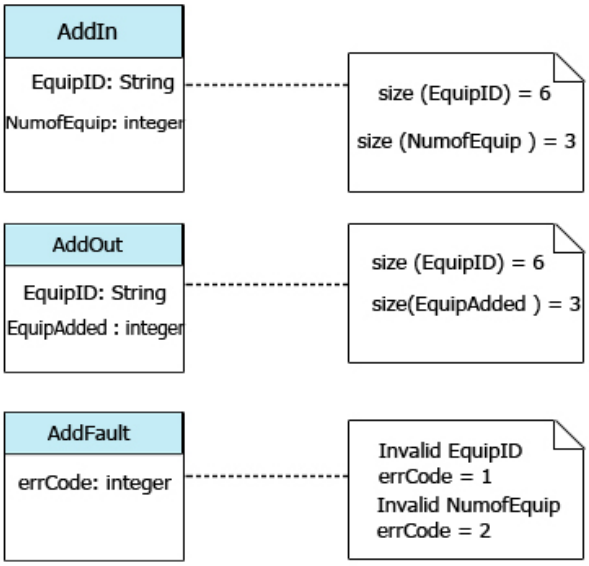


3) CompleteJob operation

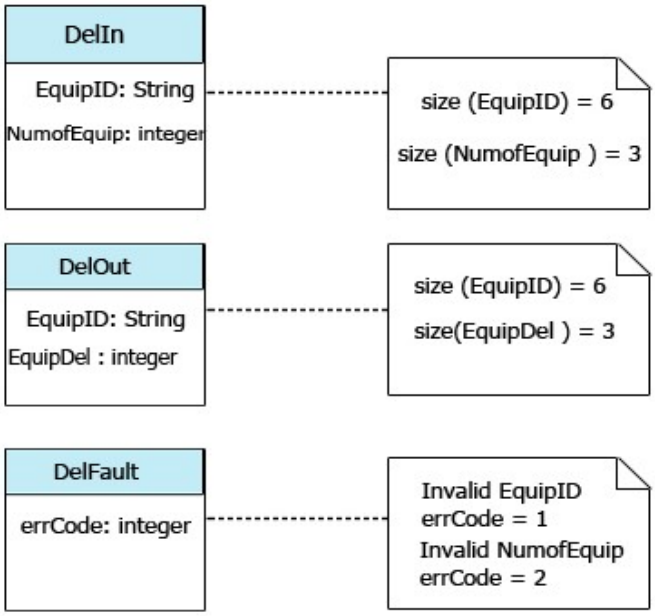


Input and output messages for Facilities service in UML notation:

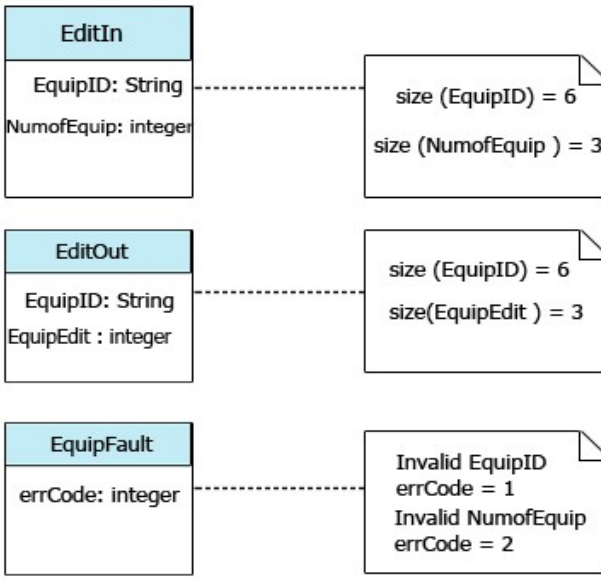
1) addEquipment operation



deleteEquipment operation

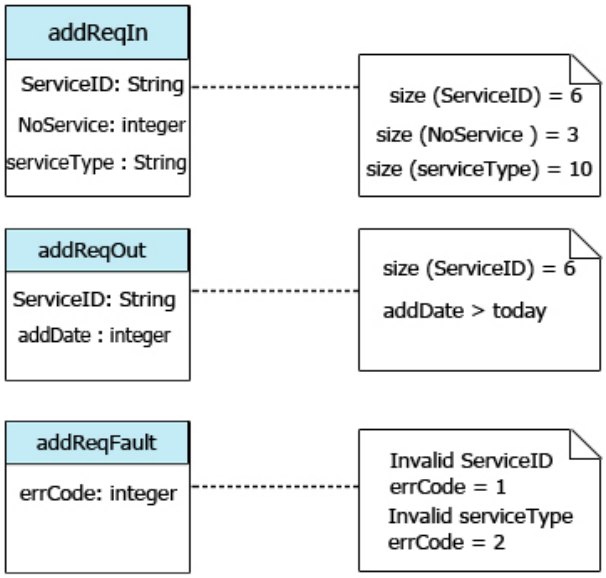


3) editEquipment operation

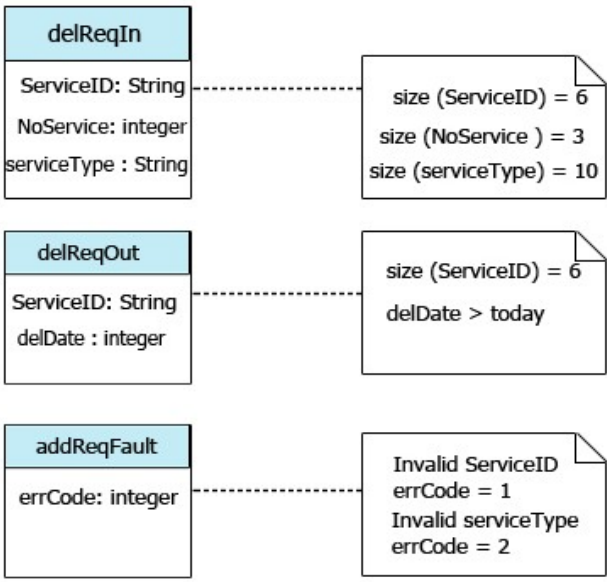


Input and output messages for Logging service in UML notation

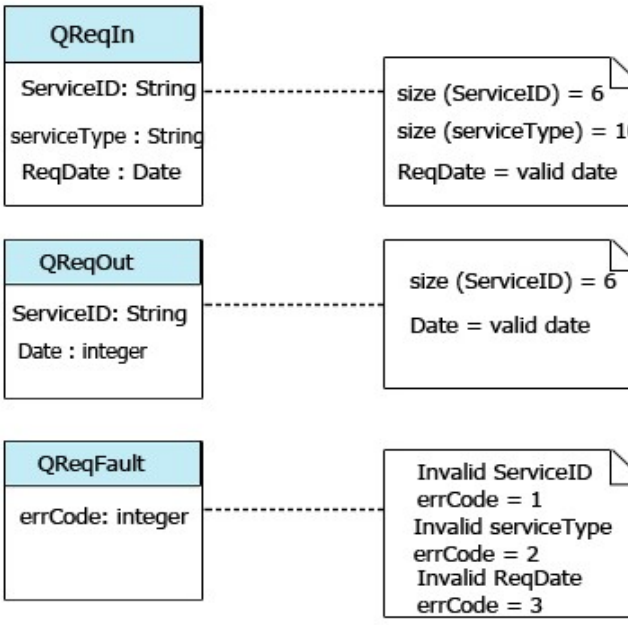
1) addRequest operation



2) deleteRequest operation



3) queryRequest operation



18.7. Giving reasons for your answer, suggest two important types of application where you would not recommend the use of service-oriented architecture.

Applications in which service-oriented architecture is not recommended:

1. Embedded applications in devices where a network connection cannot be guaranteed. These are unlikely to make use of services as there is no guarantee that the service will be available when needed
2. Applications that need GUI based functionality. Such applications are not suited for heavy data exchange that is service based.

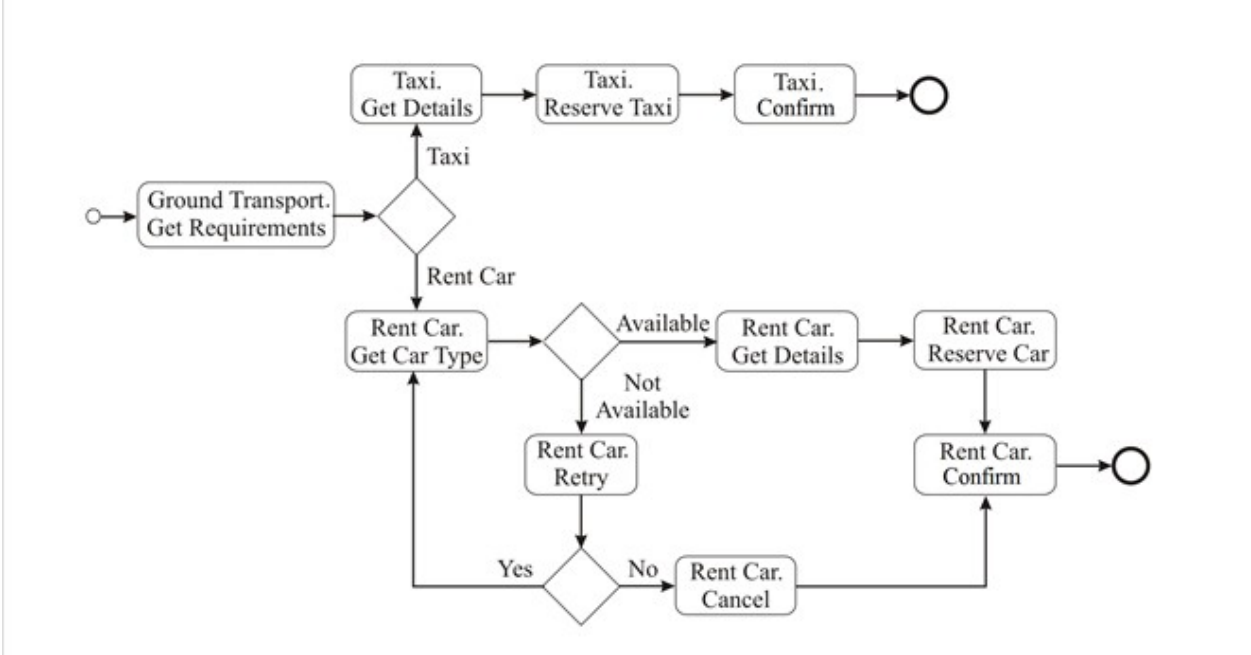
18.8. Explain what is meant by a "compensation action" and, using an example, show why these actions may have to be included in workflows.

Compensation action is an action that is used to undo actions that have already been completed but which must be changed as a result of later workflow activities. Compensating actions may have to be included in workflows because the success of the entire workflow may rely on all included workflows successfully completing.

For example, say a flight is booked to leave on 1 June and return on 7 June. The workflow then proceeds to the hotel booking stage. When no hotel rooms are available, the hotel booking service reports this lack of availability. Lack of availability is not a failure but a common situation. Therefore, then they have to undo the flight booking and pass the information about lack of availability to the user, so they can plan for alternatives.

18.9. For the example of the vacation package reservation service, design a workflow that will book ground transportation for a group of passengers arriving at an airport. They should be given the option of booking either a taxi or a hire car. You may assume that the taxi and rental car companies offer web services to make a reservation.

Workflow diagram for transportation



18.10. Using an example, explain in detail why the thorough testing of services that include compensation actions is difficult.

Compensation action is an action that is used to undo actions that have already been completed but which must be changed as a result of later workflow activities. Compensating actions may have to be included in workflows because the success of the entire workflow may rely on all included workflows successfully completing.

For example, say a flight is booked to leave on 1 June and return on 7 June. The workflow then proceeds to the resort booking stage. When no resort rooms are available, the resort booking service reports this lack of availability. Lack of availability is not a failure but a common situation. Therefore, then they have to undo the flight booking and pass the information about lack of availability to the user, so they can plan for alternatives. Therefore due to unavailability we have to undo the flight booking or change the resort. This action is called compensation action.
In this case, the booking of flight ticket depends on the availability of resort. Testing such actions is difficult as they may depend on the failure of services. The source code of the external services may not be available. In such cases, guiding the testing process become more difficult. Hence testing of services, including compensation actions is difficult.

Chapter 19

19.1. Give two examples of government functions that are supported by complex sociotechnical systems and explain why, in the foreseeable future, these functions cannot be completely automated.

Two examples of government functions are Health related services and Department of home affairs. As long as such systems provide services to different types of human users with backgrounds, capabilities, and personalities, these functions cannot be completely automated.
Automation is possible when an information system follows some pattern of work and interacts with a closed group of people. Then the tasks done by the people of

the group can be replaced by an automated system. In the case of the given examples of sociotechnical systems, automation is not possible as the sociotechnical systems interact with multiple large groups of people from various backgrounds, personalities and capabilities. Automation of the behavior of such a large number of uses from diverse backgrounds and capabilities is not possible.

19.2. Explain briefly why the involvement of a range of professional disciplines is essential in systems engineering.

An important difference between systems and software engineering is the involvement of a range of professional disciplines throughout the lifetime of the system. These include engineers who may be involved in hardware and software design, system end-users, managers who are concerned with organizational issues and experts in the system's application domain. The involvement of a range of professional disciplines is essential because of the different types of components in complex systems.

19.3. Complex sociotechnical systems lead to three important characteristics. What are they?
Explain each in brief.

This complexity leads to three important characteristics of sociotechnical systems:

1. They have emergent properties that are properties of the system as a whole, rather than associated with individual parts of the system. Emergent properties depend on both the system components and the relationships between them. Some of these relationships only come into existence when the system is integrated from its components so the emergent properties can only be evaluated at that time. Security and dependability are examples of important emergent system properties.
2. They are non-deterministic. This means that when presented with a specific input, they may not always produce the same output. The system's behaviour depends on the human operators and people do not always react in the same way. Furthermore, use of the system may create new relationships between the system components and hence change its emergent behaviour.
3. The system's success criteria are subjective rather than objective. The extent to which the system supports organizational objectives does not just

depend on the system itself. It also depends on the stability of these objectives, the relationships and conflicts between organizational objectives and how people in the organization interpret these objectives. New management may re-interpret the organizational objectives that a system was designed to support so that a 'successful' system may then be seen as no longer fit for its intended purpose.

19.4. What is a "wicked problem"? Explain why the development of a national medical records system should be considered a "wicked problem."

A wicked problem is a problem that is so complex and which involves so many related entities that there is no definitive problem specification. Different stakeholders see the problem in different ways and no one has a full understanding of the problem as a whole. The true nature of the problem may only emerge as a solution is developed.

Development of a national medical record system is an example of a wicked system:

1. There are conflicting goals of the national medical record system. One of the primary goals of the system is to improve the quality of care given to patients with the help of the medical records of the patients.
2. Another goal is to provide detailed information for managerial tasks sot that the cost of the treatments can be optimized
3. A nurse or a doctor needs to study the records of a patient closely and examine a patient to deliver better treatment to the patient, hence it will take considerable time of the doctors and nurses to fulfill the first goals.
4. At the same time, the doctors and the nurses need to spend more time in entering data into the system to provide details of the treatments for managerial tasks and to fulfill the second goals, so if the first goals is to be fulfilled then it is difficult to fulfill the second goal and vice versa
5. The problem definitions are not clear from the stakeholders. Some of the primary stakeholders of the national medical record system are the doctors, nurses and the managers.

6. For managers, the system will be helpful as it helps them in managerial tasks. For the nurses and the doctors, the system is a failure as it has put a burden of additional treatment data entry work on them.

So, due to the conflicts in the goals of the system and different options of the stakeholders, the problem of development of national medical record system can be said a wicked problem.

19.5. A multimedia virtual museum system offering virtual experiences of ancient Greece is to be developed for a consortium of European museums. The system should provide users with the facility to view 3-D models of ancient Greece through a standard web browser and should also support an immersive virtual reality experience. Develop a conceptual design for such a system, highlighting its key characteristics and essential high-level requirements.

Virtual Reality mechanisms shows a real object view experience of current objects. Each frame is a 3-D original shape of a specific object.
Conceptual design is the very first thing that you do in the systems engineering process. In the conceptual design phase, you take that initial idea, investigate its feasibility, and develop it to create an overall vision of a system that could be developed. You then have to describe the envisaged system so that nonexperts, such as system users, senior company decision makers, or politicians, can understand what you are proposing.

19.6. Explain why you need to be flexible and adapt system requirements when procuring large off-the-shelf software systems, such as ERP systems. Search the web for discussions of the failures of such systems and explain, from a sociotechnical perspective, why these failures occurred. A possible starting point

is: http://blog.360cloudsolutions.com/blog/bid/94028/ Top-Six-ERP-Implementation-Failures

19.7. Why is system integration a particularly critical part of the systems development process?
Suggest three sociotechnical issues that may cause difficulties in the system integration process.

System integration: The components are put together to create a new system. Only then do the emergent system properties become apparent. You take the independently developed subsystems put them together to make a complete system. The integration can be done using a "big-bang" approach, where all the subsystems are integrated at the same time. For technical and managerial reasons, an incremental process where subsystems are integrated one at a time is the best approach:

1. It is usually impossible to schedule the development of the subsystems so that they are all finished at the same time
2. Incremental integration reduces the cost of error location. If many subsystems are simultaneously integrated, an error that arises during testing may be in any of these subsystems
3. When a single subsystem is integrated with an already working system. Errors that occur are probably in newly integrated subsystem or in the interactions between the existing subsystems and the new subsystem.

The distinction between implementation and integration is increasingly blurred. In some cases, there is no need to develop new hardware or software and the integration is, essentially the implementation phase of the system.
Subsystem faults that are a consequence of invalid assumptions about other subsystems are often revealed during system integration. This may lead to disputes between the contractors responsible for implementing different subsystems. When problems are discovered in subsystem interaction, the contractors may argue about which subsystem is faulty.

Many difficulties can arise during deployment. The user environment may be different from that anticipated by the system developers and adapting the system to cope with diverse user environments can be difficult. The existing data may require extensive cleanup and parts of it may be missing. The interfaces to other systems may not be properly documented.

19.8. Why is system evolution inherently costly?

System evolution, like software evolution, is inherently costly for several reasons:

1. Proposed changes have to be analyzed very carefully from a business and a technical perspective. Changes have to contribute to the goals of the system and should not simply be technically motivated.
2. Because subsystems are never completely independent, changes to one subsystem may have side-effects that adversely affect the performance or behaviour of other subsystems. Consequent changes to these subsystems may therefore be needed.
3. The reasons for original design decisions are often unrecorded. Those responsible for the system evolution have to work out why particular design decisions were made.
4. As systems age, their structure becomes corrupted by change so the costs of making further changes increases.

19.9. What are the arguments for and against considering system engineering as a profession in its own right, like electrical engineering or software engineering?

Arguments for considering system engineering as a profession in its own right:

- It helps in reducing the number of employees and the manual effort
- All the categories of disciplines and their knowledge can be found under a single roof

- A person with the knowledge of all the disciplines can work well with better views

Arguments against considering system engineering as a profession in its own right:

- It will be very difficult to learn all the aspects included in the systems engineering process and get expensive
- Increases the effort in learning several issues in various disciplines
- Cannot

19.10. You are an engineer involved in the development of a financial system. During installation, you discover that this system will make a significant number of people redundant. The people in the environment deny you access to essential information to complete the system installation. To what extent should you, as a systems engineer, become involved in this situation? Is it your professional responsibility to complete the installation as contracted? Should you simply abandon the work until the procuring organization has sorted out the problem?

As a systems engineer, the completion of installation is part of his job and he as to involve in the situation only to some extent of explaining people about the benefits of the system and his duty of installation etc.
It is the professional responsibility of the systems engineer to complete the installation in normal conditions but in this type of situation where there are no supporting conditions, it is not his professional responsibility.