Solution Manual for Introduction to Software Engineering

R. Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

February 16, 2006

Chapter 1

INTRODUCTION

1. What is the principal aim of the software engineering discipline? What does the discipline of software engineering discuss?
   Ans: The principal aim of the software engineering discipline is to produce high quality software in a cost-effective manner.

2. Distinguish between a program and a software product.
   Ans: Programs are developed by individuals for their personal use. They are, therefore, small in size and have limited functionality. Further, since the author of a program himself uses and maintains his programs, these usually lack good user-interface and proper documentation. For example, the programs developed by a student as part of his class assignments are programs and not software products. On the other hand, software products have multiple users and, therefore, have good user-interface, proper users' manuals, and good documentation support. Since, a software product has a large number of users, it is systematically designed, carefully implemented, and thoroughly tested. In addition, a software product consists not only of the program code but also of all associated documents such as requirements specification document, design document, test document, users' manuals, etc. A further difference is that the software products often are too large to be developed by any individual. It is usually developed by a group of engineers working in a team.

3. Do you agree with the following statement: "The emphasis of exploratory programming is error correction while the software engineering practices emphasize error prevention"? Give the reasonings behind your answer.

Ans: Yes. Exploratory programming involves quickly coming up with a solution code and correcting it until it meets the requirements. On the other hand, software engineering emphasizes careful planning and designing and preventing errors.

4. What are the symptoms of the present software crisis? What factors have contributed to the making of the present software crisis? What are possible solutions to the present software crisis?
Ans: The symptoms of the present software crisis are increasing cost of software products, unreliable products that are full of bugs, and products delivered late.

   The factors contributing to software crisis are the increasing complexity of software products being developed, lack of professionals adequatwely trained in software engineering techniques, and lack of progress of the software engineering discipline itself.

   The possible solutions to the presenty software crisis are adequate training in software engineering techniques and progress of the software engineering discipline itself.

5. List the major differences between the exploratory and modern software development practices.
Ans:

   - Use of life cycle model emphasises phase containment of errors.
   - Modern development starts with a complete and documented requirements specification.
   - Proper design using some well-defined design methodology.
   - Systematic testing
   - Better visibility of design and code
   - Several metrics used
   - Projects are being thoroughly planned
   - Use of several CASE tools
   - Emphasis has shifted from error correction to error prevention.

6. What do you understand by control flow structure of a program? Why is it difficult to understand a program having a messy control flow structure?

Ans: The control flow structure of a program represents the sequence in which the various statements of the program are executed.

It is difficult to understand a program having a messy flow structure, since one normally understands a program by tracing different paths from the inputs to the outputs or from the outputs to the inputs. A program having a messy flow structure has an extremely large number of paths and one needs considerably longer time to trace all the paths to understand the program.

7. What is a flow chart? How is the *flow charting* technique useful for software development?
   Ans:

   - Flow charts represent flow of control in a program.
   - Flow charts can be used to design a program having good control flow structure.
   - Flow charts can be useful for documentatio and help understanding a program.

8. What do you understand by 'visibility' of design and code? How does increased visibility help in systematic software development?
   Ans: Visibility refers to the production of good quality reviewed documents. Increased visibility helps in systematic developments since it becomes easier to track the progress vis-a-vis the plans.

9. What do you understand by the term "structured programming"? How do modern programming languages such as PASCAL and C facilitate writing structured programs? What are the advantages of writing structured programs vis-a-vis unstructured programs?
   Ans: Structured programming refers to the following two important features:

   - Modular program
   - Use of only structured constructs (sequence, selection, and iteration)

   The advantage of writing structured programs is that structured programs are easier to understand and therefore are easier to debug and maintain.

10. What are the three basic types of program constructs necessary to develop the program for any given problem? Give examples of these constructs from any high-level language you know.

Ans: The three basic types of program constructs necessary to develop a program are:

- Sequence
- Selection
- Iteration

Examples of these constructs from "C" language:

- Sequence: a=b;
- Selection: if(a≤b) b=0 else c=0;
- Iteration: while(a≤b) a++;

11. What do you understand by a "program module"? What are the important characteristics of a program module.
Ans: A program module is an independently compilable piece of program code.

12. What is the basic difference between a control flow-oriented and a data flow-oriented design technique? Can you think of any reason as to why a data flow-oriented design technique is likely to produce better designs than a control flow-oriented design technique?
Ans: In a control flow-oriented design, the software is designed by carefully designing the control flow in the program using a flow chart or a similar notation.

In a data flow-oriented design, the design is arrived at from an analysis of the data flow inherent in the problem.

Data flow-oriented design techniques provide better decomposition of the problem.

13. What are the two well-known principles used in software engineering to tackle the complexity of development of large programs? Discuss these two principles and explain how these two principles help tackle program complexity.
Ans: The two well-known principles to handle complexity of development of large programs are abstraction and decomposition.

14. Discuss the possible reasons behind supersession of the data structure-oriented design methods by the control flow-oriented design methods.

15. What is a data structure-oriented software design methodology? How is it different from the data flow-oriented design methodology?
Ans: In data structure-oriented design, the data structures required in the program are first designed carefully. Based on the data structures, the code structure is designed.

16. Discuss the major advantages of the Object-Oriented Design (OOD) methodologies over the data flow-oriented design methodologies.
Ans: Object-oriented methodologies provide better abstraction and decomposition of a problem and therefore can more easily handle complexity. OOD methodologies also facilitate reuse of existing software components.

17. What is computer systems engineering? How is it different from software engineering?
Ans: Computer systems engineering addresses the issues in development of high quality systems cost-effectively.

Chapter 2

SOFTWARE LIFE CYCLE MODELS

1. What do you understand by the term "life cycle model of software development"? Why is it important to adhere to a life cycle model during development of a large software product?
   Ans: Life cycle model of software development refers to development of software through a sequence of activities structured into the phases of a life cycle model.

   It is important to follow a life cycle model in product development since software products are developed by teams of professionals. Since the product is developed with close cooperation of the team unless each team member follows the same life cycle model, it would be extremely difficult to develop the product. Also, unless a life cycle is followed software project management becomes very difficult since the project manager cannot determine the state of the project with any accuracy.

2. What problems is a software development organization likely to face if it has not adequately documented its software process?
   Ans: It becomes extremely difficult to understand and maintain a software product that is inadequately documented. Also, a development project may run into severe difficulties when any developer leaves the organization and his work is not adequately documented. Finally, unless good quality documents are produced after every milestone and these are reviewed , it becomes very difficult to track the progress of the software development.

3. What do you mean by a software process? What is the difference between a methodology and a process? What problems would a software development house face if it does not follow any process in its software development efforts?
   Ans: A software development process defines the order in which different development activities are carried out by a development team. On the other hand, a methodology spans a single phase of development whereas process spans the entire life cycle.

   If a software development house does not follow any process then many of its development efforts may end up being failures. The cost of development would be higher and management of the software development would be very difficult with large schedule slippages.

4. Which are the major phases in the waterfall model of software development? Which phase consumes the maximum effort for developing a typical software product?
   Ans: The different phases of the waterfall model model are: feasibility study, requirements analysis and specification, design, coding and unit testing, integration and system testing, and maintenance.

   Maintenance phase consumes the maximum effort among all phases. However, among the development phases testing typically requires the maximum effort.

5. Draw a schematic diagram to represent the iterative waterfall model of software development. On your diagram represent the deliverables produced at the end of each phase.
   Ans:

6. What are the important activities that are carried out during the feasibility study phase?
   Ans: During the feasibility study stage, the following activities are carried out:

   - An abstract problem definition. An abstract problem definition is a rough description of the problem which considers only the important requirements and ignoring the rest.
   - Formulation of the different solution strategies.
   - Analysis of alternate solution strategies to examine their benefits and shortcomings.

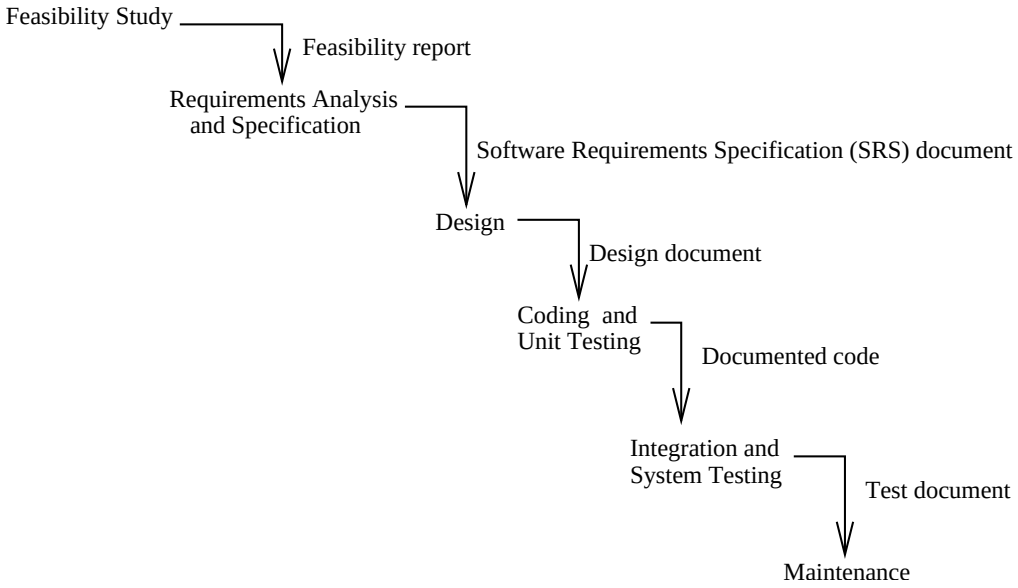

Figure 2.1: Classical waterfall model annotated with the documents produced during different phases

- Determination of whether it is technically feasible and economically worthwhile to develop the product.

7. Give examples of software product developments for which the iterative waterfall model is not suitable.
   Ans: A product for which the customer requirement is not well-defined or a product or one that suffers from many risks.

8. Discuss how the effort spent in the different phases of the iterative waterfall model are spread over time.
   Ans: Completion of each phase typically requires relatively different amounts of effort to be put in by the development team. The relative effort necessary for completing the activities for different phases for a typical product is shown in Fig. 2.

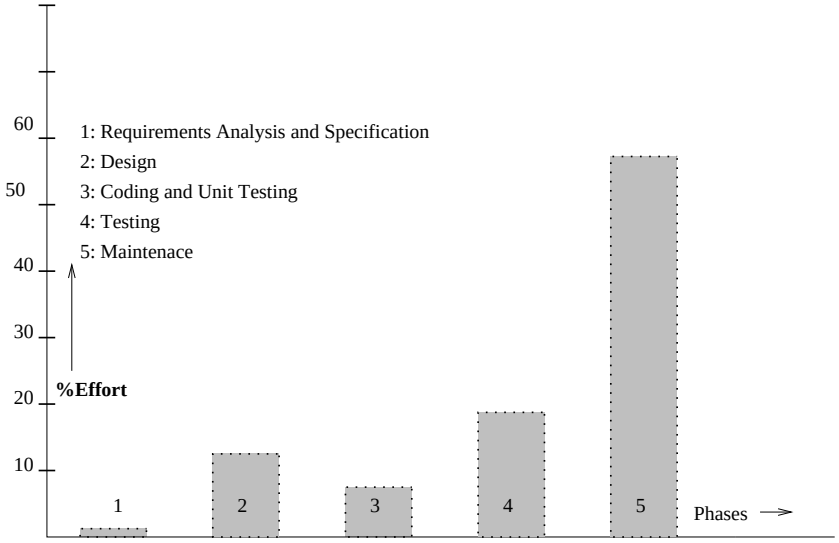

Figure 2.2: Relative effort distribution among different phases of a typical product

9. Which life cycle model would you follow for developing software for each of the following applications? Mention the reasons behind your choice of a particular life cycle model.

(a) A well-understood data processing application.
Ans: Iterative waterfall model. It is the most cost-effective model of development for well-understood problems.

(b) A new software product that would connect computers through satellite communication. Assume that your team has no previous experience in developing satellite communication software.
Ans: Prototyping model can be followed, since the team members have no previous experience in developing satellite communication software.

(c) A software product that would function as the controller of a telephone switching system.
Ans: A prototyping model or an iterative waterfall model may be chosen based on the familiarity of the development team in developing similar products.

(d) A new library automation software that would link various libraries in the city.
Ans: A prototyping model or an iterative waterfall model may be chosen based on the familiarity of the development team in developing similar products.

(e) An extremely large software that would provide, monitor, and control cellular communication among its subscribers using a set of revolving satellites.
Ans: Spiral model or evolutionary model can be followed. Since, the product is large, either an evolutionary model or a spiral model may be chosen based on the familiarity of the development team in the development of similar products.

(f) A new text editor.
Ans: A prototyping model or an iterative waterfall model may be chosen based on the familiarity of the development team in developing similar products.

(g) A compiler for a new language.
Ans: A prototyping model or an iterative waterfall model may be chosen based on the familiarity of the development team in developing similar products.

(h) An object-oriented software development effort.
Ans: Evolutionary model may be chosen. Since objects make up well-defined components of a product, a evolutionary model leads to a more manageable and reliable product.

(i) The graphical user interface part of a large software product.
Ans: Prototyping model may be chosen, since the users often

change their mind regarding various features of the user interface.

10. What is a prototype? Under what circumstances is it beneficial to construct a prototype? Does construction of a prototype always increase the overall cost of software development?
Ans: A prototype is a working model of a system. It is beneficial to construct a prototype either when the requirements are not well defined, or there are some technical risks in the development.

    No. Development of a throw away prototype does not always increase the cost of development, since the cost of construction of a prototype is more than offset by the reworks that might have to be done in presence of ill-defined user requirements and technical risks.

11. What are the major advantages of first constructing a working prototype before developing the actual product?
Ans: A prototype helps eliminate ambiguities in user requirements and also can be used to eliminate technical risks. Also, the experience in developing a prototype helps to develop a better product during the actual implementation.

12. Explain how a software development effort is initiated and finally terminated in the spiral model.
Ans: In the spiral model, during each phase (i.e, loop of the spiral) some goal is identified. The risks in this goal is eliminated through the construction of a prototype and finally the goal is implemented. The product development is complete when the risks in all the goals have been resolved and all goals have been implemented.

13. Explain why the spiral life cycle model is considered to be a meta model.
Ans: The spiral life cycle model is considered to be a meta model, because the spiral model subsumes all other development models. a single loop spiral actually represents the waterfall model. The spiral model uses a prototyping approach by first building a prototype before the actual product development starts. Also, the spiral model can be considered as supporting the evolutionary model — the iterations along the spiral can be considered as evolutionary levels through which the complete system is built. This enables the developer to understand and resolve the risks at each evolutionary level (i.e., iteration along the spiral). The spiral model uses prototyping as a risk reduc-

tion mechanism and also retains the systematic step-wise approach of the waterfall model.

14. Explain with suitable examples, the types of software development for which the spiral model is suitable. Is the number of loops of the spiral fixed for different development projects? If not, explain how the number of loops in the spiral is determined.
Ans: Spiral model is suitable for very large software development efforts that are subject to many kinds of risks.

    No, the number of loops of the spiral are not fixed. The loops in the spiral are determined based on the number of risks that the product is susceptible to.

15. How are the risks associated with a project handled in the spiral model of software development?
Ans: Risks are handled in a spiral model through construction of prototypes.

16. Compare the relative advantages of using the iterative waterfall model and the spiral model of software development. Explain with the help of a few suitable examples, the type of problems for which you would adopt the waterfall model of software development, and the type of problems for which you would adopt the spiral model.
Ans: Iterative waterfall model is a simple model to follow and is very efficient for well understood problems.

17. Explain using suitable examples the type of product developments for which the evolutionary life cycle is model and the problems for which the spiral model is suitable.
Ans: Evolutionary model is suitable for develop of products that require very large effort and investment and also for products where a basic working system with minimal feature is immediately required. For example, when a product is to be developed quickly to capture the market, an evolutionary model may be used to quickly release a basic working model to the market.

18. State whether the following statements are **TRUE** or **FALSE**. Give reasons for your answer.

    (a) The primary purpose of *phase containment of errors* is to develop an error-free product.
    Ans: FALSE. The primary purpose of phase containment of errors is to reduce the cost of error correction.

(b) Development of a software product using the prototyping life cycle model is always more expensive than development of the same product using the iterative waterfall model, since additional cost is incurred to construct a throw-away prototype.
Ans: FALSE. Development of a throw away prototype does not always increase the cost of development, since the cost of construction of a prototype is more than offset by the reworks that might have to be done in presence of ill-defined user requirements and technical risks.

(c) Software development organizations which follow the iterative waterfall model for product development, provide maximum customer satisfaction.
Ans: FALSE. The customers often are unsatisfied as they do not see the product until the development is complete.

(d) Among all phases of software development, an undetected error from the design phase which ultimately gets detected during the system acceptance test costs the maximum.
Ans: FALSE. A requirements error that gets detected in the system acceptance test costs the maximum.

19. Explain why it is not prudent to use the iterative waterfall model for developing very large software products.
Ans: It is not prudent to use iterative model may not be prudent for very large product development efforts as the development organization may not have as many development personnel and resources or the customer may not be able to pay upfront for the entire product. Also, large products are often subject to more number of risks.

20. Instead of having a one time testing of a software product at the end of its development, why are three different levels of testing — unit testing, integration testing, and system testing — necessary? What is the main purpose of each of these different levels of testing?
Ans:

21. What do you understand by the term *phase containment of errors*? Why is phase containment of errors is important? How can phase containment of errors be achieved?
Ans:

- The principle of detecting errors as close to their points of introduction as possible is known as *phase containment of errors*.
- Phase containment of errors reduces the cost of correction of errors.

- Phase containment of errors can be achieved through thorough reviews.

22. Irrespective of whichever life cycle model is followed for a software product development, why is it necessary for the final documents to describe the product as if it were developed using the classical waterfall model?
Ans: If the final document is written to reflect a classical waterfall model of development, comprehension of the system documents becomes easy. The rationale behind this argument is explained by CAR Hoare using the metaphor of mathematical theorem proving. A mathematician presents a proof as a single chain of deductions, even though the proof might have come from a convoluted set of partial attempts, blind alleys and backtracks.

23. Suppose you plan to undertake the development of a product beset with a large number of technical as well as customer related risks, which life cycle model would you adopt? Give the reasonings behind your answer.
Ans: A spiral model or prototyping model can be used depending on the size of the product.

24. Suggest a suitable life-cycle model for a software project which your organization has undertaken on behalf of certain customer who is likely to change his requirements frequently. Give the reasonings behind your answer.
Ans: A prototyping model would be suitable. The prototype can be changed until the customer freezes his requirements and after that the development can be started.

Chapter 3

SOFTWARE PROJECT MANAGEMENT

1. List the major responsibilities of a software project manager.
   Ans: The major responsibilities of a project manger includes project proposal writing, project cost estimation, scheduling, project staffing, software process tailoring, project monitoring and control, software configuration management, risk management, interfacing with clients, managerial report writing and presentations, etc. These responsibilities can be broadly divided into project planning and project monitoring and control activities.

2. What do you under stand by sliding window planning? Explain using a few examples the types of projects for which this form of planning is especially suitable. What are its advantages over conventional planning?
   Ans: In the sliding window technique, starting with an initial plan, the project is planned more accurately in successive development stages. At the start of a project, project managers have incomplete knowledge about the details of the project. Their information base gradually improves as the project progresses through different phases. After the completion of every phase, the project managers can plan each subsequent phase more accurately and with increasing levels of confidence.

   For large projects, conventional planning turns out to be too inaccurate. Sliding window planning is used for large projects to improve the accuracy and effectiveness of planning.

3. When does the project planning activity start and end in a software life cycle? List the important activities software project managers perform during project planning.
   Ans: Project planning starts after the project is found to be feasible and continues till the end of testing phase. The important activities that managers perform during project planning are:

   - Estimating some basic attributes of the project.

     Cost: How much is it going to cost to develop the project?

     Duration: How long is it going to take to complete development?

     Effort: How much effort would be required?
   - Scheduling manpower and other resources
   - Staff organization and staffing plans
   - Risk identification, analysis, and abatement planning
   - Miscellaneous plans such as quality assurance plan, configuration management plan, etc.

4. What do you understand by product visibility in the context of software development? Why is it important to improve product visibility during software development?
   Ans: Product visibility implies development of high quality reviewed documents at the end of every milestone.

   Since software is invisible, it is very difficult to manage the development. Increased visibility helps manager to effectively monitor progress and control the project.

5. What are the different categories of software development projects according to the COCOMO estimation model? Give examples of software product development projects belonging to each of these categories.
   Ans: The different categories of software development projects according to the COCOMO estimation model are:

   Organic: Data processing applications.

   Semidetached: Utility programs such as compilers, computer communication software, etc.

   Embedded: These require programming at the hardware level such as embedded system development.

Besides the inherent complexity, the other factors that affect the classification of a project into the different categories of development are the experience and proficiency of the development team.

6. For the same number of lines of code and the same development team size, rank the following software projects in order of their estimated development time. Briefly mention the reasoning behind your answer.

   - A text editor
   - An employee pay roll system
   - An operating system for a new computer

   Ans:

   (a) An employee pay roll system

   (b) A text editor

   (c) An operating system for a new computer

   The reason for the chosen ordering is that the three projects can be respectively classified as: Organic,semidetached, and embedded according COCOCMO.

7. As the the manager of a software project to develop a product for business application, if you estimate the effort required for completion of the project to be 50 person-months, can you complete the project by employing 50 engineers for a period of one month? Justify your answer.
   Ans: No, it would not be possible to complete the project by employing 50 engineers for a period of one month. If 50 engineers are hired for 1 month, then it would be difficult to meaningfully keep each of them busy for the month because of the limited parallel activities and the ordering among the tasks. Therefore the project duration and cost would increase.

8. State whether the following statements are **TRUE** or **FALSE**. Give reasons for your answer.

(a) As a project manager it would be worthwhile on your part to reduce the project duration by half provided the customer agrees to pay for the increased manpower requirements.
Ans: FALSE
Because of the limited parallel activities and the ordering among the tasks it would not be possible to decrease the development time any less than 75% of the nominal estimated time.

(b) Software organizations achieve efficient manpower utilization by adopting project-based organization structure.
Ans: FALSE
Functional organizations achieve efficient manpower utilization, since at any time the required man power can be borrowed and returned to the functional pool. Also, a functional organization leads to job specialization.

(c) For the development of the same product, the larger is the size of a software development team, the faster is the product development. (for simplicity, assume that all engineers are equally proficient and have exactly similar experience).
Ans:FALSE
Because of the limited parallel activities and the ordering among the tasks it would not be possible to decrease the development time any less than 75% of the nominal estimated time whatever may be the number of manpower employed.

(d) The number of development personnel required for any software development project can be obtained by dividing the total (estimated) effort by the total (estimated) duration of the project.
Ans: FALSE
A simple division of total (estimated) effort by the total (estimated) duration implicitly assumes constant team size. But, the team size should show a Raleigh distribution for efficient utilization of the manpower.

(e) The democratic team organization is very well suited to handle complex and challenging projects.
Ans: TRUE
Democratic set up encourages thinking by all team members and is suited to handle complex and challenging projects.

(f) It is possible to do the configuration management of a software project without using an automated tool.
Ans: TRUE
It would be difficult to do the configuration management of a

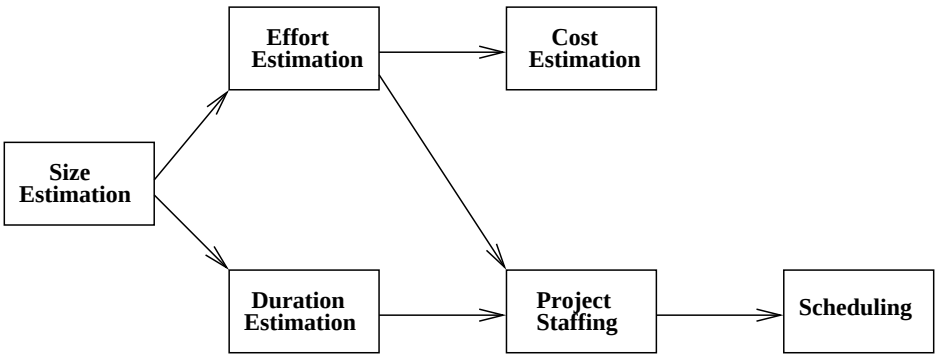

Figure 3.1: Precedence ordering among planning activities

software project without using an automated tool, but not entirely impossible.

(g) According to the COCOMO model, cost is the most fundamental attribute of a software product, based on which size and effort are estimated.
Ans: FALSE
According COCOMO model size is the most fundamental attribute of a software product, based on which effort and cost are estimated.

(h) Size of a project, as used in COCOMO is the size of of the final executable code in bytes.
Ans: FALSE
Size as used in COCOMO is the measure of the problem complexity in terms of the effort and time required to develop the product.

9. Using a schematic diagram show the order in which the following are estimated in the COCOMO estimation technique: cost, effort, duration, size.
Ans: The precedence ordering is shown in Figure 1.

10. Suppose you have estimated the nominal development time of a moderate-sized software product to be 5 months. You have also estimated that it will cost Rs.50,000/- to develop the software product. Now, the customer comes and tells you that he wants you to accelerate the delivery time by 10%. How much additional cost would you charge the

customer for this accelerated delivery? Irrespective of whether you take less time or more time to develop the product, you are essentially developing the same product. Why then does the effort depend on the duration over which you develop the product?
Ans: Since the product is moderate-sized, we can use the Jensen Formula.
new cost = 50,000*(5/4)*(5/4)= Rs. 78,125/-

The cost increases because limited amounts of parallel activities are possible in every project. Employing more people than the nominal estimate it would not be possible to keep all engineers fully engaged with meaningful work. The idle time of personel shows up as increased cost.

11. Explain how Putnam's model can be used to compute the change in project cost with project duration. What are the main disadvantages of using the Putnam's model to compute the additional costs incurred due to schedule compression? How can you overcome them?
Ans:

$$K_1/K_2 = t_{d_2}^4/t_{d_1}^4$$

From this expression, it can be easily observed that when the schedule of a project is compressed, the required effort increases in proportion to the fourth power of the degree of compression. It means that a relatively small compression in delivery schedule can result in substantial penalty on human effort. For example, if the estimated development time is 1 year, then in order to develop the product in 6 months, the total effort required to develop the product (and hence the project cost) increases 16 times.

Putnam estimation model works reasonably well for very large systems, but seriously overestimates the effort on medium and small systems. For medium and small projects, Jensen model may be used.

12. Suppose you are developing a software product in the organic mode. You have estimated the size of the product to be about 100,000 lines of code. Compute the nominal effort and the development time.
Ans:

Given that the size is 100 KLOC and the project is organic. Nominal effort=$2.4 \times 100^{1.05}$ = 2.4*125.893=302.1 man-months
Nominal development time =$2.5 \times (Effort)^{0.38}$ = $2.5 \times 302.1^{0.38}$=8.6 months

13. For the following C program estimate the Halstead's length and volume measures. Compare Halstead's length and volume measures of

size with the LOC measure.

```
/* Program to calculate GCD of two numbers */
        int compute_gcd(x,y)
            int x,y;
        {
                while (x != y)
                   if (x>y) then x=x-y;
                   else y=y-x;
                return x;
        }
```

14. What does Halstead's volume metric represent conceptually? How according to Halstead is the effort dependent on program volume?
Ans: Volume represents the size of the program by approximately compensating for the effect of the programming language used. According to Halstead the effort to develop a program varies as the square of the volume and is given as V^*/V .

15. What are the relative advantages of using either the LOC or the function point metric to measure the size of a software product?

16. List the important shortcomings of LOC for use as a software size metric.
Ans:

- LOC gives a numerical value of problem size that can vary widely with individual coding style.
- LOC, however, focuses on the coding activity alone; it merely computes the number of source lines in the final program.
- LOC measure correlates poorly with the quality and efficiency of the code.
- LOC metric penalizes use of higher-level programming languages, code reuse, etc.
- LOC metric measures the lexical complexity of a program and does not address the more important but subtle issues of logical or structural complexities.
- It is very difficult to accurately estimate LOC in the final product from the problem specification.

17. Explain why adding more man power to an already late project makes it later.
Ans: The new personnel must understand the work already completed before they can be productive, this requires time of the existing members.

18. The following table indicates the various tasks involved in completing a software project, the corresponding activities, and the estimated effort for each task in person-months.

Notation	Activity	Effort in person-months
T_1	Requirements specification	1
T_2	Design	2
T_3	Code actuator interface module	2
T_4	Code sensor interface module	5
T_5	Code user interface part	3
T_6	Code control processing part	1
T_7	Integrate and test	6
T_8	Write user manual	3

The precedence relation $T_i \leq \{T_j, T_k\}$ implies that the task T_i must complete before either task T_j or T_k can start. The following precedence relation is known to hold among different tasks: $T_1 \leq T_2 \leq \{T_3, T_4, T_5, T_6\} \leq T_7$. Draw the **Activity network** and the **Gantt chart** representations for the project.
Ans:
See Fig 2.

19. Suppose you are the project manager of a software project requiring the following activities.

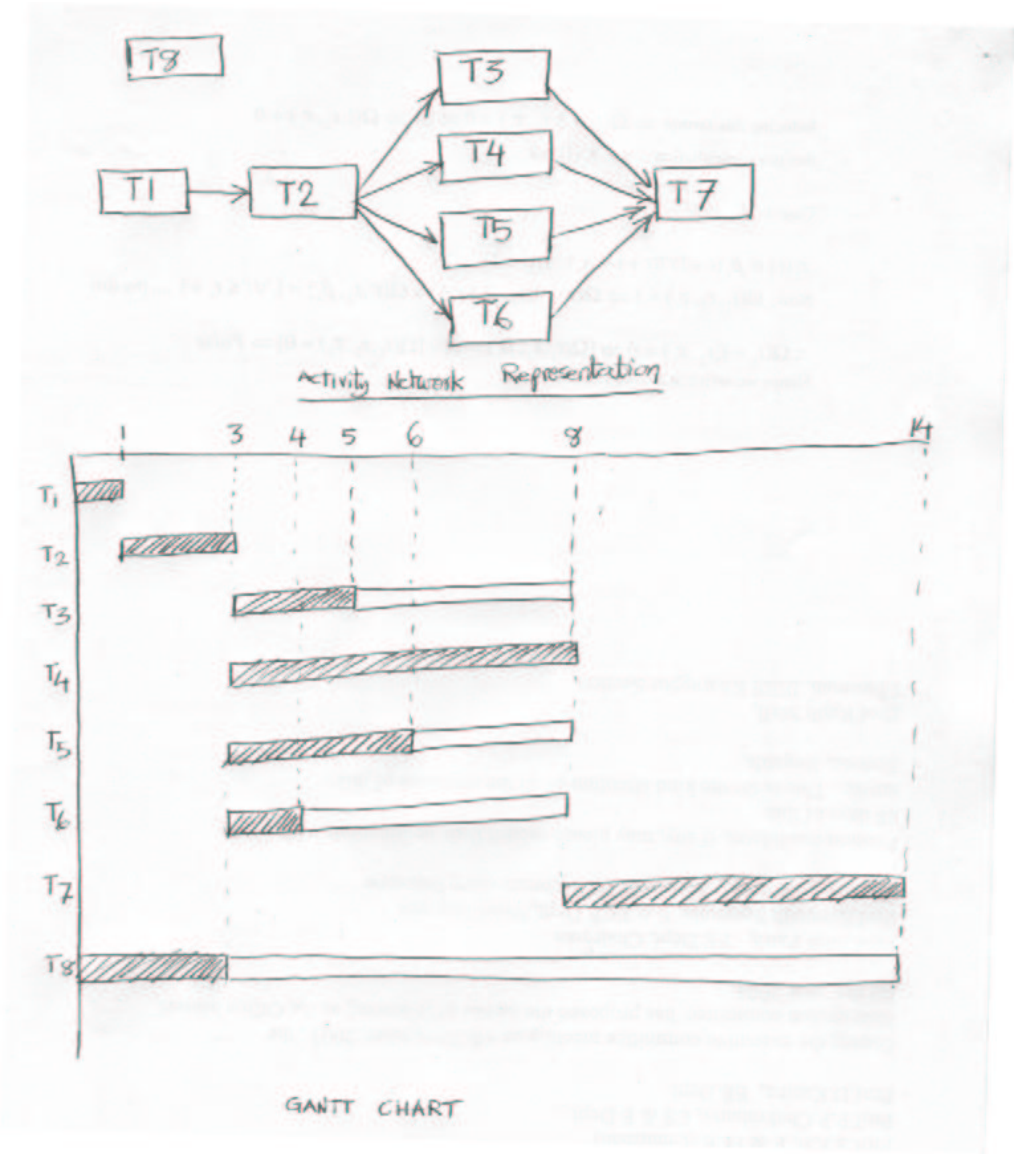

Figure 3.2: Activity diagram and Gantt Chart for problem 18.

Activity No.	Activity Name	Duration (weeks)	Immediate Predecessor
1.	Obtain requirements	4	-
2.	Analyze operations	4	-
3.	Define subsystems	2	1
4.	Develop database	4	1
5.	Make decision analysis	3	2
6.	Identify constraints	2	5
7.	Build module 1	8	3,4,6
8	Build module 2	12	3,4,6
9.	Build module 3	18	3,4,6
10.	Write report	10	6
11.	Integration and test	8	7,8,9
12.	Implementation	2	10,11

(a) Draw the Activity Network representation of the project.

(b) Draw the Gantt chart representation of the project.

20. Explain when you should use PERT charts and when you should use Gantt charts while you are performing the duties of a project manager.

21. Suppose that software product for business application costs Rs. 50,000/- to buy off-the-shelf and that its size is 40 KLOC. Assuming that in-house engineers cost Rs.6000/- per programmer-month (including overheads), would it be more cost-effective to buy the product or build it? Which elements of the cost are not included in COCOMO estimation model? What additional factors should be considered in making the buy/build decision?
Ans: The product is for business application and can be classified as Organic type. The cost according to the COCOMO model is therefore,
Nominal effort=$2.4 \times 40^{1.05}$ = 2.4*48.1=115.5 man-months

In-house engineers cost Rs.6000/-. So, the cost is 115.5*6000=Rs. 692,669/-

However, before a buy or build decision is made, the following points must be taken into account.

- How quickly is the product required? It may take some time to develop the product.

- Number of copies required.
- Whether the developed product can be sold?
- Whether any maintenance of the product be required in the future and what would be the cost of the maintenance?
- The risks associated with development such as schedule slippage, cost escalation, etc.
- Would the experience gained in the product development be of use?

22. Suppose you are the manager of a software project. Explain using only one or two sentences why you should not calculate the number of engineers required for the project as a simple division of the effort estimate (in person-months) by the nominal duration estimate (in months)?
Ans: A simple division of total (estimated) effort by the total (estimated) duration implicitly assumes constant team size. But, the team size should show a Raleigh distribution for efficient utilization of the manpower. So, a constant team size arrived from a simple division of the effort estimate (in person-months) by the nominal duration estimate (in months) would lead to wastage of manpower and consequent increase of cost and duration.

23. List the important items that a Software Project Management Plan (SPMP) document should discuss.
Ans:

24. Suppose you are the project manager of a large product development team and you have to make a choice between democratic and chief programmer team organizations, which one would you adopt for your team? Explain the reasoning behind your answer.
Ans:

(a) **Introduction**
(b) **Project Estimates**
(c) **Schedule**
(d) **Project Resources**
(e) **Staff Organization**
(f) **Risk Management Plan**
(g) **Project Tracking and Control Plan**
(h) **Miscellaneous Plans**

25. Compare the relative advantages of the functional and the project approaches of development center organization. Suppose you are the chief executive officer (CEO) of a software development center. Which organization structure would you select for your organization? Why?

26. Name the different ways in which software development teams are organized. For the development of a challenging satellite-based mobile communication product which type of project team organization would you recommend? Justify your answer.
Ans: The team organization for the challenging satellite-based mobile communication product would depend on the size of the product being developed. If the size of the product is small, then a democratic organization would be suitable. Otherwise, a mixed organization can be used.

27. What do you understand by software configuration? What is meant by software configuration management? How can you manage software configuration (only mention the names of the principal activities involved)? Why is software configuration management crucial to the success of large software product development projects (write only the important reasons)?
Ans: The state of all deliverable items at any point of time is called the *configuration* of the software product. Software configuration management involves effectively managing the configuration. Software configuration can be manged through:

    - **Configuration identification**
    - **Configuration control**

28. In what units can you measure the productivity of a software development team? List three important factors that affect the productivity of a software development team.
Ans: The productivity of a development team can be measured in units of developed KLOC per day. Three important factors that affect the productivity of a development team are:

    - Use of software engineering principles
    - Use automated tools
    - Experience and motivation of the team members

29. List three common types of risks that a typical software project might suffer from. Explain how you can identify the risks that your project

is susceptible to. Suppose you are the project manager of a large software development project, point out the main steps you would follow to manage risks in your software project.
Ans: Three common types of risks that a typical software project might suffer from:

- Schedule slippage
- Frequent requirements change by the customer
- Team members leaving the organization

The main steps in managing risks are:

- Risk Identification
- Risk Assessment
- Risk Containment
  - **Avoid the risk:**
  - **Transfer the risk:**
  - **Risk reduction:**

30. Schedule slippage is a very common form of risk that almost every project manager has to encounter. Explain in 3 to 4 sentences how you would manage the risk of schedule slippage as the project manager of a medium-sized project.
Ans: Schedule slippage can be handled by setting regular milestones, producing good quality documents at the completion of the mile stone and having rigorous review of the documents.

31. Explain how you can choose the best risk reduction technique when there are many ways of reducing a risk.
Ans: To choose between the different strategies of handling a risk, the project manager must consider the cost of handling the risk and the corresponding reduction of risk. For this we may compute the **risk leverage** of the different risks. Risk leverage is the difference in risk exposure divided by the cost of reducing the risk. More formally,

$$risk\ leverage = \frac{risk\ exposure\ before\ reduction\ -\ risk\ exposure\ after\ reduction}{cost\ of\ reduction}$$

32. What are the important types of risks that a project might suffer from? How would you identify the risks that a project is susceptible to during project the project planning stage?
Ans: The different types of risks that a project may suffer can be broadly classified into:

Project risks.

Technical risks.

Business risks.

In order to be able to successfully foresee and identify different risks that might affect a software project, it is a good idea to have a **company disaster list**. This list would contain all the bad events that have happened to software projects of the company over the years including events that can be laid at the customer's doors. This list can be read by the project mangers in order to be aware of some of the risks that a project might be susceptible to. Such a disaster list has been found to help in performing better risk analysis.

33. As a project manager, identify the characteristics that you would look for in a software engineer while trying to select personnel for your team.
Ans: The following characteristics are especially important:

    - Exposure to systematic techniques, i.e. familiarity with software engineering principles.
    - Good technical knowledge of the project areas (*Domain knowledge*)
    - Good programming abilities
    - Good communication skills. These skills comprise of oral, written, and interpersonal skills.
    - High motivation
    - Sound knowledge of fundamentals of computer science
    - Intelligence
    - Ability to work in a team
    - Discipline, etc.

34. Discuss how SCCS or RCS can be used to efficiently manage the configuration of source code.
Ans: The change control facilities provided by SCCS and RCS include the ability to incorporate restrictions on the set of individuals who can create new versions, and facilities for checking components in and out (i.e. reserve and restore operations). Also, SCCS and RCS provide an efficient way of storing versions that minimizes the amount of occupied disk space.

35. What is *egoless programming*? How can it be realized?
Ans: An egoless programming team thinks that the design, code, and other deliverables to belong to the entire group rather that different parts of the code owned by different individuals of the team. This is called *egoless programming* because it tries to avoid having programmers invest much ego in the development activity they do in a democratic set up.

36. Is it true that a software product can always be developed faster by having a larger development team (you can assume that all engineers are equally proficient and have exactly similar experience)? Justify your answer.
Ans: Because of the limited parallel activities and the ordering among the tasks it would not be possible to decrease the development time any less than 75% of the nominal estimated time.

37. Suppose you have been appointed as the project manager of a large project, identify the activities you would undertake to plan your project. Explain the sequence in which you would undertake these activities by using a flow chart notation. What are some of the factors which make it hard to accurately estimate the cost of software projects?

38. Suppose you are the project manager of a large development project. The top management informs that you would have to do with a fixed team size (i.e. constant number of engineers) through out the duration your project. What will be the impact of this decision on your project?
Ans: The project may be delayed (schedule slippage), the project cost may escalate, etc.

39. The industry average productivity figure for engineers is only 10 LOC/day. What is the reason for such low productivity? Can we attribute this to the poor programming skill of engineers?
Ans: The productivity is not really as low as it appears since coding is only part of the activity and lots of effort goes into designing, testing etc. which are not reflected in the LOC/day productivity measure.

40. What problems would you face if you are developing several versions of the same product according to a client's request, and you are not using any configuration management tools?
Ans: The following problems might arise:

Inconsistency problem when the objects are replicated.

Problems associated with concurrent access.

Providing a stable development environment.

System accounting and maintaining status information.

Handling variants.

41. What is the difference between a revision and a version? What do you understand by the terms change control and version control? Why are these necessary? Explain how change and version control are achieved using a configuration management tool.
Ans: A new version of a software is created when there is significant change in functionality, technology, or the hardware it runs on, etc. On the other hand, a new release is created if there is only a bug fix, minor enhancements to the functionality, usability, etc.

Change control refers to managing the changes that occurs to the configuration of a system as development proceeds and also when new revisions are produced. Version control deals with managing multiple versions efficiently.

Chapter 4

Requirements Analysis and Specification

1. Suppose you have been appointed as the analyst for a large software development project. Discuss the aspects of the software product you would document in the Software Requirements Specification (SRS) document? What would be the organization of your SRS document?
   Ans:
   The following aspects would be included:
   (a) Introduction
   (b) Functional Requirements
   (c) Non-functional Requirements

2. Using suitable examples explain the different types of requirements problems that should be identified and resolved during the requirements analysis activity.
   Ans:
   The important types of requirements problems that should be identified and resolved during the requirements analysis activity are: ambiguity, incompleteness, and inconsistency.

3. List five desirable characteristics of a *good* Software Requirements Specification (SRS) document.
   Ans:
   Some of the identified desirable qualities of the SRS documents are the following:
   - **Concise.**

- **Structured.**
- **Black-box view.**
- **Conceptual integrity.**
- **Response to undesired events.**
- **Verifiable.**

4. Suppose the analyst of a large product development effort has prepared the SRS document in the form of a narrative essay of the system to be developed. Based on this document, the product development activity gets underway. Explain the problems that such a requirements specification document may create during development?
   Ans:
   The following problems would be encountered:

   - It would be very difficult to modify any requirements later on if the SRS document is written in the form of a narrative essay. Simple changes may require modifications at several places.
   - It would be very difficult to remove ambiguity, imprecision, incompleteness, and contradiction.
   - It would be very difficult to trace the requirements with the design, code, test documents, etc.

5. Discuss the relative advantages of formal and informal requirements specifications.
   Ans:

   - Formal specifications encourage rigour.
   - Formal specifications are not only more precise, but also mathematically sound and can be used to reason about the properties of a specification.
   - Formal methods have well-defined semantics. Therefore, ambiguity in specifications is automatically avoided when one formally specifies a system.
   - The mathematical basis of the formal methods facilitates automating the analysis of specifications.
   - Informal specifications are easier to understand.

6. Why is the SRS document also known as the black-box specification of a system?

Ans:
An SRS document describes the externally visible behavior of the system without referring to the internals of the systems.

7. Who are the different category of users of the SRS document? What are their expectations from the SRS document?
Ans:
Some of the important categories of users of the SRS document and their needs are as follows:

Users,customers, and marketing personnel. The goal of this set of audience in referring to the SRS document is to ensure that the system as described will meet their needs.

Software developers: The software developers refer to the SRS document to make sure that they are developing exactly what is required by the customer.

Test engineers: Their goal is to ensure that the requirements are understandable from a functionality point of view, so that they can test the software and validate its working. they need that the functionality be clearly described, and the input and output data identified precisely.

User documentation writers: Their goal in reading the SRS document is to ensure that they understand the document well enough to be able to write the users' manuals.

Project managers: They want to ensure that they can estimate the cost easily by referring to the SRS document and that it contains all the information required to plan the project well.

Maintenance engineers: The SRS document helps the maintenance engineers to understand the functionality of the system. A clear knowledge of the functionality can help them to understand the design and code. Also, the requirements knowledge would enable them to determine what modifications to the system's functionality would be needed for a specific purpose.

8. What do you understand by traceability of requirements? Why is traceability important?
Ans:
Traceability means that it would be possible to tell which design component corresponds to which requirement, which code corresponds to which design component, and which test case corresponds to which requirement, etc. Thus, a given code component can be traced to

the corresponding design component, and a design component can be traced to a specific requirement it implements and vice versa. Traceability analysis is an important concept and is frequently used during software development. For example, by doing a traceability analysis, we can tell whether all the requirements have been duly taken into account in all phases. It can also be used to assess the impact of a requirements change. That is, traceability makes it easy to identify which parts of the design and code would be affected, when any requirement changes. It can also be used to study the impact of a bug on various requirements.

9. State whether the following statements are **TRUE** or **FALSE**. Give reasons for your answer.

   (a) Applications developed using 4GLs would normally be more efficient and run faster compared to applications developed using 3GL.
   Ans:
   FALSE
   4GL implementations lead to very general solutions where common abstractions across all similar applications have been parameterized. This makes it inefficient.

   (b) A formal specification cannot be ambiguous.
   Ans:
   TRUE
   Since formal specification languages have well-defined semantics, ambiguity is automatically avoided.

   (c) A formal specification cannot be incomplete.
   Ans:
   FALSE It is possible that a formal specification is incomplete.

   (d) A formal specification cannot be inconsistent.
   Ans:
   FALSE It is possible that a formal specification is inconsistent. However, the inconsistency can be automatically detected.

   (e) The system test plan can be prepared immediately after the completion of the requirements specification phase.
   Ans:
   TRUE
   Since the system test plan is designed solely based on the functional and nonfunctional specification of the system, it can be prepared immediately after the specification phase.

10. (a) What are the primary differences between a model-oriented specification method and a property-oriented specification method.
Ans:
The specification of a system can be given either as a list of its desirable properties (property-oriented approach) or as an abstract model of the system (model-oriented approach).

    (b) Compare the relative advantages of property-oriented specification methods over model-oriented specification methods.
Ans:
Property-oriented approaches specify a system by a conjunction of axioms, thereby making it easier to alter/augment specifications at a later stage. On the other hand, model-oriented methods do not support logical conjunctions and disjunctions, and thus even minor changes to a specification may lead to overhauling an entire specification.

    (c) Name at least one representative popular property-oriented specification technique, and one representative model-oriented specification technique.
Ans:
Property-oriented specification: axiomatic and algebraic specification styles. Model-oriented specification: FSM, Petri nets.

11. Consider the following requirement for a software to be developed for controlling a chemical plant. The chemical plant has a number of emergency conditions. When any of the emergency conditions occurs, some prespecified actions should be taken. The different emergency conditions and the corresponding actions that need to be taken are as follows:

    (a) If the temperature of the chemical plant exceeds $T_1^\circ C$, then the water shower should be turned ON and the heater should be turned OFF.

    (b) If the temperature of the chemical tank falls below $T_2^\circ C$, then the heater should be turned ON and the water shower should be turned OFF.

    (c) If the pressure of the chemical plant is above P_1, then the valve v_1 should be OPENED.

    (d) If the chemical concentration of the tank rises above M, and the temperature of the tank is more than $T_3^\circ C$, then the water shower should be turned ON.

(e) If the pressure rises above P_3 and the temperature rises above $T_1^\circ C$, then the water shower should be turned ON, valves v_1 and v_2 are OPENED and the alarm bells sounded.

Write the requirements of this chemical plant software in the form of a decision table.

12. Draw a decision tree to represent the processing logic of the previous problem.

13. Represent the decision making involved in the operation of the following wash-machine by means of a decision table:
The machine waits for the `start` switch to be pressed. After the user presses the `start` switch, the machine fills the wash tub with either hot or cold water depending upon the setting of the `HotWash` switch. The water filling continues until the high level is sensed. The machine starts the agitation motor and continues agitating the wash tub until either the preset timer expires or the user presses the `stop` switch. After the agitation stops, the machine waits for the user to press the `startDrying` switch. After the user presses the `startDrying` switch, the machine starts the hot air blower and continues blowing hot air into the drying chamber until either the user presses the `stop` switch or the preset timer expires.

14. Represent the processing logic of the following problem in the form of a decision table:
A Library Membership Automation System needs to support three functions: **add new-member, renew-membership, cancel-membership**. If the user requests for any function other than these three, then an error message is is flashed. When an **add new-member** request is made, a new member record is created and a bill for the annual membership fee for the new member is generated. If a membership renewal request is made, then the expiry date of the concerned membership record is updated and a bill towards the annual membership fee is generated. If a membership cancellation request is made, then the concerned membership record is deleted and a cheque for the balance amount due to the member is printed.

15. What do you understand by pre- and post-conditions of a function? Write the pre- and post-conditions to axiomatically specify the following functions:

(a) A function takes two floating point numbers representing the sides of a rectangle as input and return the area of the corre-

sponding rectangle as output.

(b) A function accepts three integers in the range of -100 and +100 and determines the largest of the three integers.

(c) A function takes an array of integers as input and finds the minimum value.

(d) A function named square-array creates a 10 element array where each all elements of the array, the value of any array element is square of its index.

(e) A function sort takes an integer array as its argument and sorts the input array in ascending order.

16. Using the algebraic specification method, formally specify a **string** supporting the following operations:
    - **append**: append a given string to another string
    - **add**: add a character to a string
    - **create**: create a new null string
    - **isequal**: checks whether two strings are equal or not
    - **isempty**: checks whether the string is null
17. Using the algebraic specification method, formally specify an **array** of generic type elem. Assume that array supports the following operations:
    - **create** takes the array bounds as parameters and initializes the values of the array to undefined.
    - **assign** creates a new array, where a particular element has been assigned a value.
    - **eval** reveals the value of a specified element.
    - **first** returns the first bound of the array.
    - **last** returns the last bound of the array.
18. What do you mean by an executable specification language? Name a commercially available executable specification language.
    Ans:

If the specification of a system is expressed formally or by using a programming language, then it becomes possible to directly execute the specification. Examples of commercially available executable specification languages are SQL. YACC, LEX.

19. What is a *fourth generation* programming technique? What are its advantages and disadvantages vis-a-vis a third generation technique?
Ans:
Using a 4GL only the "what" parts have to be specified. (4th Generation Languages) are examples of executable specification languages. 4GLs are successful because there is a lot of commonality across data processing applications. 4GLs rely on software reuse, where the common abstractions have been identified and parameterized. Careful experiments have shown that rewriting 4GL programs in higher level languages results in up to 50% lower memory usage and also the program execution time can reduce ten folds.

20. What are auxiliary functions in algebraic specifications? Why are these needed?
Ans:
Auxiliary functions are extra functions (not part of the interface functionality of the system) that are introduced to to define the meaning of some interface procedures.

21. What do you understand by incremental development of algebraic specifications? What is the advantage of incremental development of algebraic specifications?
Ans:
The idea behind incremental specification is to first develop the specifications of the simple types and then specify more complex types by using the specifications of the simple types.

22. (a) Algebraically specify an abstract data type that stores a **set** of elements and supports the following operations. Assume that the ADT element has already been specified and you can use it:
    i. **new:** creates a null set.
    ii. **add:** takes a set and an element and returns the set with the additional elements stored.
    iii. **size:** takes a set as argument and returns the number of elements in the set.
    iv. **remove:** takes a set and an element as its argument and returns the set with the element removed.

v. **contains:** takes a set and an element as its argument and returns the boolean value true if the element belongs to the set and returns false if the element does not belong to the set.

vi. **equals:** takes two sets as arguments and returns true if they contain identical elements and returns false otherwise.

(b) Using the specification you have developed for the ADT set, reduce the following expression by applying the rewrite rules: equals(add(5,(add(6,new()),add(6,(add(5,new())))). Show the details of every reduction.

23. Write a formal algebraic specification of the sort **symbol-table** whose operations are informally defined as follows:

- **create:** bring a symbol table into existence.
- **enter:** enter a symbol table and its type into the table.
- **lookup:** return the type associated with a name in the table.
- **delete:** remove a name, type pair from the table, given a name as a parameter.
- **replace:** replace the type associated with a given name with the type specified as its parameter.

The enter operation fails if the name is already present in the table. The lookup, delete, and replace operations fail if the name is not available in the table.

24. Algebraically specify the data type **queue** which supports the following operations:

- **create:** creates an empty queue.
- **append:** takes a queue and an item as its arguments and returns a queue with the item added at the end of the queue.
- **remove:** takes a queue as its argument and returns a queue with the first element of the original queue removed.
- **inspect:** takes a queue as its argument and returns the value of the first item in the queue.
- **isempty:** takes a queue as its argument and returns true if the queue contains no elements, and returns false if it contains one or more elements.

You can assume that the data type item has previously been specified and that you can reuse this specification.

25. Define the finite termination and unique termination properties of algebraic specifications? Why is it necessary for an algebraic specification to satisfy these properties?
Ans:
Unique termination property indicates whether application of rewrite rules in different orders always result in the same answer. Finite termination property essentially addresses the following question: Do applications of the rewrite rules to arbitrary expressions involving the interface procedures always terminate?

These properties are important to be able to get meaningful results from the application of the property.

Chapter 5

Software Design

1. What do you mean by the terms cohesion and coupling in the context of software design? How are these concepts useful in arriving at a good design of a system?
   Ans:
   Cohesion is a measure of the functional strength of a module whereas the coupling between two modules is a measure of the degree of interdependence or interaction between the two modules. Cohesion and coupling help in arriving at a modular design.

2. Enumerate the different types of cohesion that a module might exhibit.
   Ans:
   The different types of cohesion are:

   - **Coincidental cohesion.**
   - **Logical cohesion.**
   - **Temporal cohesion.**
   - **Procedural cohesion.**
   - **Communicational cohesion.**
   - **Sequential cohesion.**
   - **Functional cohesion.**

3. Enumerate the different types of coupling that might exist between two modules.
   Ans:
   The different types of coupling are:

- **Data coupling.**
- **Stamp coupling.**
- **Control coupling.**
- **Common coupling.**
- **Content coupling.**

4. Is it true that whenever you increase the cohesion of different modules in your design, coupling among these modules automatically decreases? Justify your answer by using a suitable example.

5. What according to you is a good software design?
Ans:
The characteristics of a good software design are as follows: These characteristics are listed below:

   Correctness: A good design should correctly implement all the functionalities of the system.

   Understandability: A good design is easily understandable.

   Efficiency: It should be efficient.

   Maintainability: It should be easily amenable to change.

6. What do you understand by the term *functional independence* in the context of software design? What is the advantage of functional independence? How can you achieve functional independence in a software design?
Ans:
A module having high cohesion and low coupling is said to be *functionally independent* of other modules. By the term *functional independence,* we mean that a cohesive module performs a single task or function. A functionally independent module has minimal interaction with other modules. Functional independence is a key to any good design primarily due to the following reasons:

   Error isolation: Functional independence reduces error propagation. The reason behind this is that if a module is functionally independent, its degree of interaction with other modules is less. Therefore, any error existing in a module would not directly affect the other modules.

   Scope for reuse: Reuse of a module becomes possible. Because each module does some well-defined and precise function and

the interface of the module with other modules is simple and minimal. Therefore, a cohesive module can be easily taken out and reused in a different program.

Understandability: Complexity of the design is reduced, because different modules can be understood in isolation as modules are more or less independent of each other.

7. Explain why a design approach based on the information hiding principle is likely to lead to a reusable and maintainable design. Illustrate your answer with a suitable example.
   Ans:
   A design approach based on the information hiding principle is likely to lead to a reusable and maintainable design since information hiding leads to low coupling and high cohesion among the modules. Thus, the modules become functionally independent.

8. State whether the following statements are **TRUE** or **FALSE**. Give reasons for your answer.

   (a) The essence of any good function-oriented design technique is to map the functions performing similar activities into a module.
   Ans:
   FALSE
   Such a design would have low cohesion. The functions that together achieve some meaningful result should be mapped together.

   (b) Traditional design is carried out top-down whereas object-oriented design is normally carried out bottom-up.
   Ans:
   TRUE
   Traditional design is based on decomposing high-level functions into subfunctions, sub-subfunctions and so on. Whereas, object-oriented design is based on identifying the natural objects in a problem and identifying various interrelations between them and combining them to more abstract classes and so on.

9. Compare relative advantages of the object-oriented and function-oriented approaches to software design.
   Ans:
   Object-oriented techniques have gained wide acceptance because of their simplicity, code and design reuse scope they offer, promise of lower development time, lower development cost, more robust code, and easier maintenance.

10. Name a few well-established function-oriented software design techniques.
Ans:
The names of a few well-established function-oriented software design techniques are:

    - Constantine and Yourdon [1979]
    - Hatley and Pirbhai [1987]
    - Gane and Sarson [1979]
    - DeMarco and Yourdon [1978]

11. Explain some of the causes of and remedies for high coupling between two software modules.
Ans:
The causes of high coupling can be complex interfaces among the functions, use of global data, and improper distribution of functions to modules.

12. What problems are likely to arise if two modules have high coupling?
Ans:
When two modules have high coupling, it would be difficult to understand the functions of the modules and to reuse the modules. Debugging would also become difficult.

13. What problems are likely to occur if a module has low cohesion?
Ans:
It would be difficult to understand, debug, and reuse the module.

14. Distinguish between high-level and detailed designs. What documents should be produced on completion of high-level and detailed designs?
Ans:
During high-level design, the important components (modules) of the system and their interactions are identified. During detailed design, the algorithms, data structures, etc. are identified.

    On completion of the high-level design the module structure in the form of structure charts should be produced. The outcome of the detailed design stage is usually known as the *module specification* document.

15. What is meant by the term *cohesion* in the context of software design? Is it true that in a good design, the modules should have low cohesion? Why?
Ans:

Cohesion is a measure of the functional strength of a module. It would be difficult to understand, debug, and reuse the module.

16. What is meant by the term *coupling* in the context of software design? Is it true that in a good design, the modules should have low coupling? Why?
Ans:
Coupling between two modules is a measure of the degree of interdependence or interaction between the two modules.

    Yes, in a good design, the modules should have low coupling. When two modules have high coupling, it would be difficult to understand the functions of the modules and to reuse the modules. Debugging would also become difficult.

17. How would you improve a software design that displays very low cohesion and high coupling?
Ans:
By redistributing the functions to modules and redesigning the modules.

18. What do you understand by the term top-down decomposition in the context of function-oriented design?
Ans:
Traditional design is based on decomposing high-level functions into subfunctions, sub-subfunctions and so on. This is referred to as top-down design.

19. What do you understand by a layered software design? What are the advantages of a layered design?
Ans:
In a layered design solution, the modules are arranged in layers. A layered design improves understandability of the design and makes it more maintainable.

20. What is the principal difference between the software design methodologies based on functional abstraction and those based on data abstraction? Name at least one popular design technique based on each of these two software design paradigms.
Ans:
Object-oriented design techniques are based on data abstraction. Structured analysis and design technique is based on functional abstraction.

21. What are the main advantages of using a object-oriented design approach over a function-oriented approach?

Ans:
Object-oriented techniques have gained wide acceptance because of their simplicity, code and design reuse scope they offer, promise of lower development time, lower development cost, more robust code, and easier maintenance.

22. Point out three important differences between the function oriented and the object oriented approaches to software design.
Ans:

- Unlike function-oriented design methods, in OOD, the basic abstraction are not real-world functions such as *sort, display, track,* etc., but real-world entities such as *employee, picture, m achine, radar system,* etc.
- In OOD, state information is not represented in a centralized shared memory but is distributed among the objects of the system.
- Function-oriented techniques such as SA/SD group functions together if, as a group, they constitute a higher level function. On the other hand, object-oriented techniques group functions together on the basis of the data they operate on.

Chapter 6

Function-Oriented Software Design

1. What do you understand by the term "top-down decomposition" in the context of function-oriented design?
   Ans:
   The term "top-down decomposition" denotes successive decomposition of the set of high-level functions of the system into more detailed functions.

2. Distinguish between a data flow diagram (DFD) and a flow chart.
   Ans:
   A flow chart represents the transfer of control between elements, whereas data flow diagram represents the data dependency aspects.

3. Differentiate between structured analysis and structured design in the context of function-oriented design.
   Ans:
   Structured analysis is concerned with decomposing high-level functions into more detailed functions successively and representing them in the form of DFDs. Structured design, on the other hand, is concerned about transforming a DFD representation in to structure chart.

4. Point out the important differences between a structure chart and a flow chart as design representation techniques.
   Ans:
   A structure chart differs from a flow chart in three principal ways:

- It is usually difficult to identify the different modules of a program from its flow chart representation.
- Data interchange among different modules is not represented in a flow chart.
- Sequential ordering of tasks inherent in a flow chart is suppressed in a structure chart.

5. What do you mean by the term *data dictionary* in the context of structured analysis? How is the data dictionary useful in different phases of the life cycle of a software product?
   Ans:
   A data dictionary lists all data items appearing in the DFD model of a system. The data items listed include all data flows and the contents of all data stores appearing on all the DFDs in the DFD model of a system.

   A data dictionary plays a very important role in any software development process because of the following reasons:

   - A data dictionary provides a standard terminology for all relevant data for use by the engineers working in a project. A consistent vocabulary for data items is very important, since in large projects different engineers of the project have a tendency to use different terms to refer to the same data, which unnecessarily causes confusion.
   - The data dictionary provides the analyst with a means to determine the definition of different data structures in terms of their component elements.

6. State whether the following statements are **TRUE** or **FALSE**. Give reasons behind your answers.

   (a) The essence of any good function-oriented design principle is to map similar functions into a module.
   Ans:
   FALSE
   Such a design would have low cohesion. The functions that together achieve some meaningful result should be mapped together.

7. What do you mean by *balancing a DFD*? Illustrate your answer with a suitable example.
   Ans:

The data that flow into or out of a bubble must match the data flow at the next level of DFD. This is known as *balancing* a DFD.

8. What are the main shortcomings of Data Flow Diagram (DFD) as a tool for performing structured analysis?
   Ans:
   The important shortcomings of DFD models are the following:

   - DFDs leave ample scope to be imprecise. In the DFD model, we judge the function performed by a bubble from its label. However, a short label may not capture the entire functionality of a bubble.
   - Control aspects are not defined by a DFD. For instance, the order in which inputs are consumed and outputs are produced by a bubble is not specified. A DFD model does not specify the order in which the different bubbles are executed. Representation of such aspects is very important for modelling real-time systems.
   - The method of carrying out decomposition to arrive at the successive levels and the ultimate level to which decomposition is carried out are highly subjective and depend on the choice and judgment of the analyst. Due to this reason, even for the same problem, several alternative DFD representations are possible. Further, many times it is not possible to say which DFD representation is superior or preferable to another one.
   - The data flow diagramming technique does not provide any specific guidance as to how exactly to decompose a given function into its subfunctions and we have to use subjective judgment to carry out decomposition.

9. What is the significance of design reviews? Make a list of items that can be used as a checklist for carrying out design reviews.
   Ans:
   Design reviews help detect various types of design flaws. Checklist for design review:

   - Whether all functional specifications can be traced to design components and vice versa.
   - Balancing of DFDs
   - Whether all bubbles of the DFDs can be traced to modules of structure charts and vice versa.

10. (a) Draw a labelled DFD for the following time management software. Clearly show the context diagram and its hierarchical decompositions up to level 2. (Note: Context diagram is the Level 0 DFD).

Time Management Software: A company needs to develop a time management system for its executives. The software should let the executives register their daily appointment schedules. The information to be stored includes person(s) with whom meeting is arranged, venue, the time and duration of the meeting, and the purpose (e.g. for a specific project work). When a meeting involving many executives needs to be organized, the system should automatically find a common slot in the diaries of the concerned executives, and arrange a meeting (i.e. make relevant entries in the diaries of all the concerned executives) at that time. It should also inform the concerned executives about the scheduled meeting through e-mail. If no common slot is available, TMS should help the secretary to rearrange the appointments of the executives in consultation with the concerned executives for making room for a common slot. To help the executives check their schedules for a particular day the system should have a very easy-to-use graphical interface. Since the executives and the secretaries have their own desktop computers, the time management software should be able to serve several remote requests simultaneously. Many of the executives are relative novices in computer usage. Everyday morning the time management software should e-mail every executive his appointments for the day. Besides registering their appointments and meetings, the executives might mark periods for which they plan to be on leave. Also, executives might plan out the important jobs they need to do on any day at different hours and post it in their daily list of engagements. Other features to be supported by the TMS are the following: TMS should be able to provide several types of statistics such as which executive spent how much time on meetings. For which project how many meetings were organized for what duration and how many man-hours were devoted to it. Also, it should be able to display for any given period of time the fraction of time that on the average each executive spent on meetings.

(b) Using the DFD you have developed for Part (a) of this question,

develop the structured design for the time management software.

Ans:
See Fig. 1.

11. Perform structured analysis and structured design for the following software that would automate the book keeping activities of a 5-star hotel.

Hotel Automation Software: A hotel has a certain number of rooms. Each room can be either single bed or double bed type and may be AC or Non-AC type. The rooms have different rates depending on whether they are of single or double, AC or Non-AC types. The room tariff however may vary during different parts of the year depending up on the occupancy rate. For this, the computer should be able to display the average occupancy rate for a given month, so that the manager can revise the room tariff for the next month either up wards or down wards by a certain percentage.

Guests can reserve rooms in advance or can reserve rooms on the spot depending upon availability of rooms. The receptionist would enter data pertaining to guests such as their arrival time, advance paid, approximate duration of stay, and the type of the room required. Depending on this data and subject to the availability of a suitable room, the computer would allot a room number to the guest and assign a unique token number to each guest. If the guest cannot be accommodated, the computer generates an apology message. The hotel catering services manager would input the quantity and type of food items as and when consumed by the guest, the token number of the guest, and the corresponding date and time. When a customer prepares to check-out, the hotel automation software should generate the entire bill for the customer and also print the balance amount payable by him. During check-out, guests can opt to register themselves for a frequent guests program. Frequent guests should be issued an identity number which helps them to get special discounts on their bills.

Ans:
See Fig. 2.

12. Perform structured analysis and structured design (SA/SD) for a software to be developed for automating various book keeping activities of a small book shop. From a discussion with the owner of the book

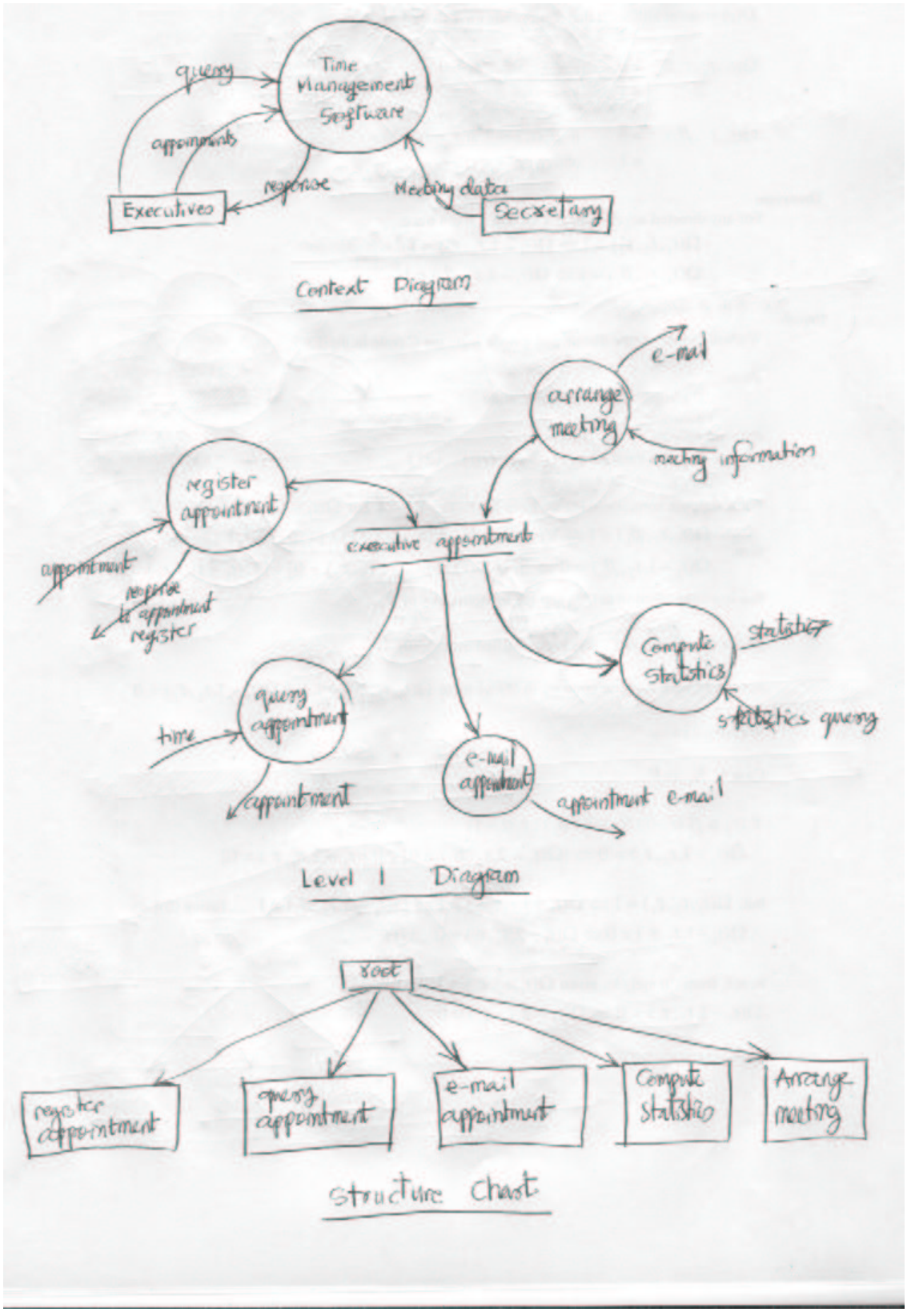

Figure 6.1: Structured Analysis and Design for TIme Management Software

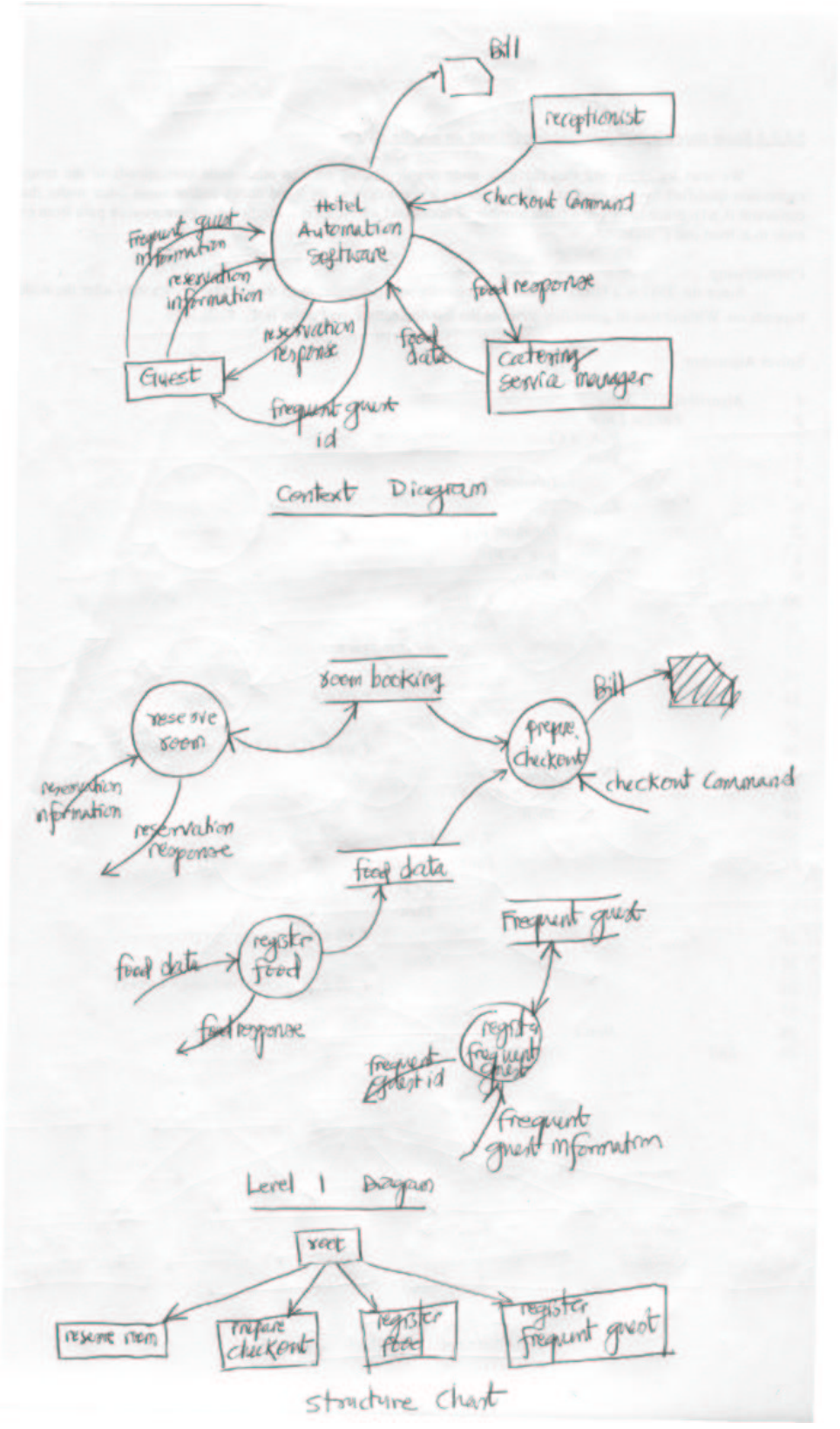

Figure 6.2: Structured Analysis and Design for Hotel Automation Software

shop, the following user requirements have been arrived at:

Book-shop Automation Software (BAS): BAS should help the customers query whether a book is in stock. The users can query the availability of a book either by using the book title or by using the name of the author. If the book is not currently being sold by the book-shop, then the customer is asked to enter full details of the book for procurement of the book in future. The customer can also provide his e-mail address, so that he can be intimated automatically by the software as and when the book copies are received. If a book is in stock, the exact number of copies available and the rack number in which the book is located should be displayed. If a book is not in stock, the query for the book is used to increment a request field for the book. The manager can periodically view the request field of the books to arrive at a rough estimate regarding the current demand for different books. BAS should maintain the price of various books. As soon as a customer selects his books for purchase, the sales clerk would enter the ISBN numbers of the books. BAS should update the stock, and generate the sales receipt for the book. BAS should allow employees to update the inventory whenever new supply arrives. Also upon request by the owner of the book shop, BAS should generate sales statistics (viz., book name, publisher, ISBN number, number of copies sold, and the sales revenue) for any period. The sales statistics will help the owner to know the exact business done over any period of time and also to determine inventory level required for various books. The inventory level required for a book is equal to the number of copies of the book sold over a period of one week multiplied by the average number of weeks it takes to procure the book from its publisher. Every day the book shop owner would give a command for the BAS to print the books which have fallen below the threshold and the number of copies to be procured along with the full address of the publisher.

Ans:
See Fig. 3.

13. Perform structured analysis and structured design for the following Road Repair and Tracking Software (RRTS) to be developed for automating various book keeping activities associated with the road repairing task of the Public Works Department of the Municipal Corporation of a large city.
Road Repair and Tracking System (RRTS): A city corporation has branch offices at different suburbs of the city. Residents

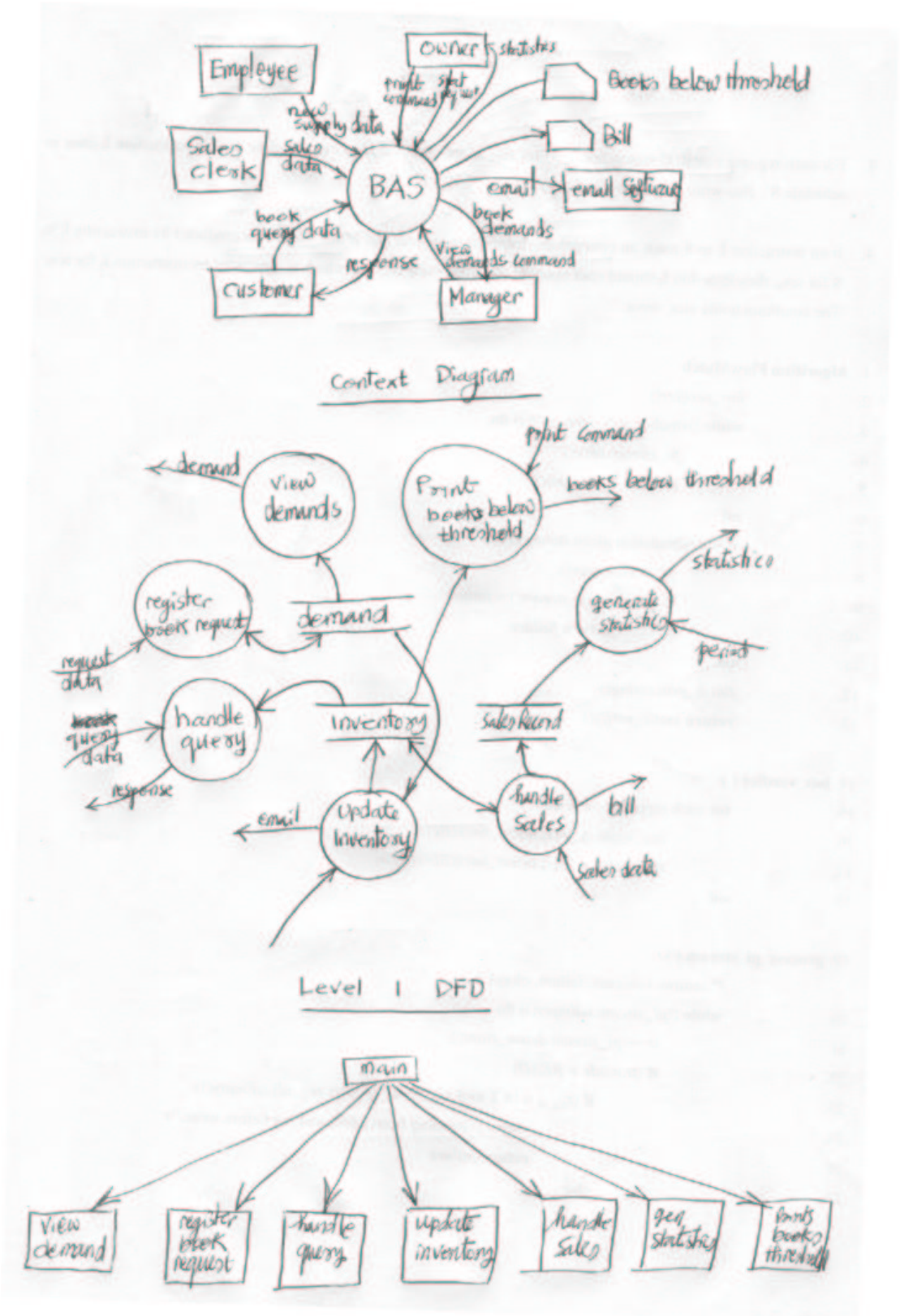

Figure 6.3: Structured Analysis and Design for Bookshop Automation Software

raise repair requests for different roads of the city. These would be entered into the computer system by a clerk. Soon after a repair request is raised, a supervisor visits the road and studies the severity of road condition. Depending on the severity of the road condition and the type of the locality (e.g., commercial area, busy area, relatively deserted area, etc.), he determines the priority for carrying out this repair work. The supervisor also estimates the raw material requirement for carrying out the repair work, the types and number of machines required, and the number and types of personnel required. Based on this data, the computer system should schedule the repair of the road depending up on the priority of the repair work and subject to the availability of raw material, machines, and personnel. This schedule report is used by the supervisor to direct different repair work. The manpower and machine that are available are entered by the city corporation administrator. He can change these data any time. Of course, any change to the available manpower and machine would require a reschedule of the projects. The mayor of the city can request for various road repair statistics such as the number and type of repairs carried out over a period of time and the repair work outstanding at any point of time and the utilization statistics of the repair manpower and machine over any given period of time.

Ans:
See Fig. 4.

14. Perform structured analysis and structured design for developing the following Restaurant Automation System using the SA/SD technique.
Restaurant Automation System (RAS): A restaurant owner wants to computerize his order processing, billing, and accounting activities. He also expects the computer to generate statistical report about sales of different items. A major goal of this computerization is to make supply ordering more accurate so that the problem of excess inventory is avoided as well as the problem of non-availability of ingredients required to satisfy orders for some popular items is minimized. The computer should maintain the prices of all the items and also support changing the prices by the manager. Whenever any item is sold, the sales clerk would enter the item code and the quantity sold. The computer should generate bills whenever food items are sold. Whenever ingredients are issued for preparation of food items, the data is to be entered into the computer. Purchase orders are generated on a daily basis, whenever the stock for any ingredient falls below a threshold value. The computer should calculate the threshold value for each item based on the average consumption of

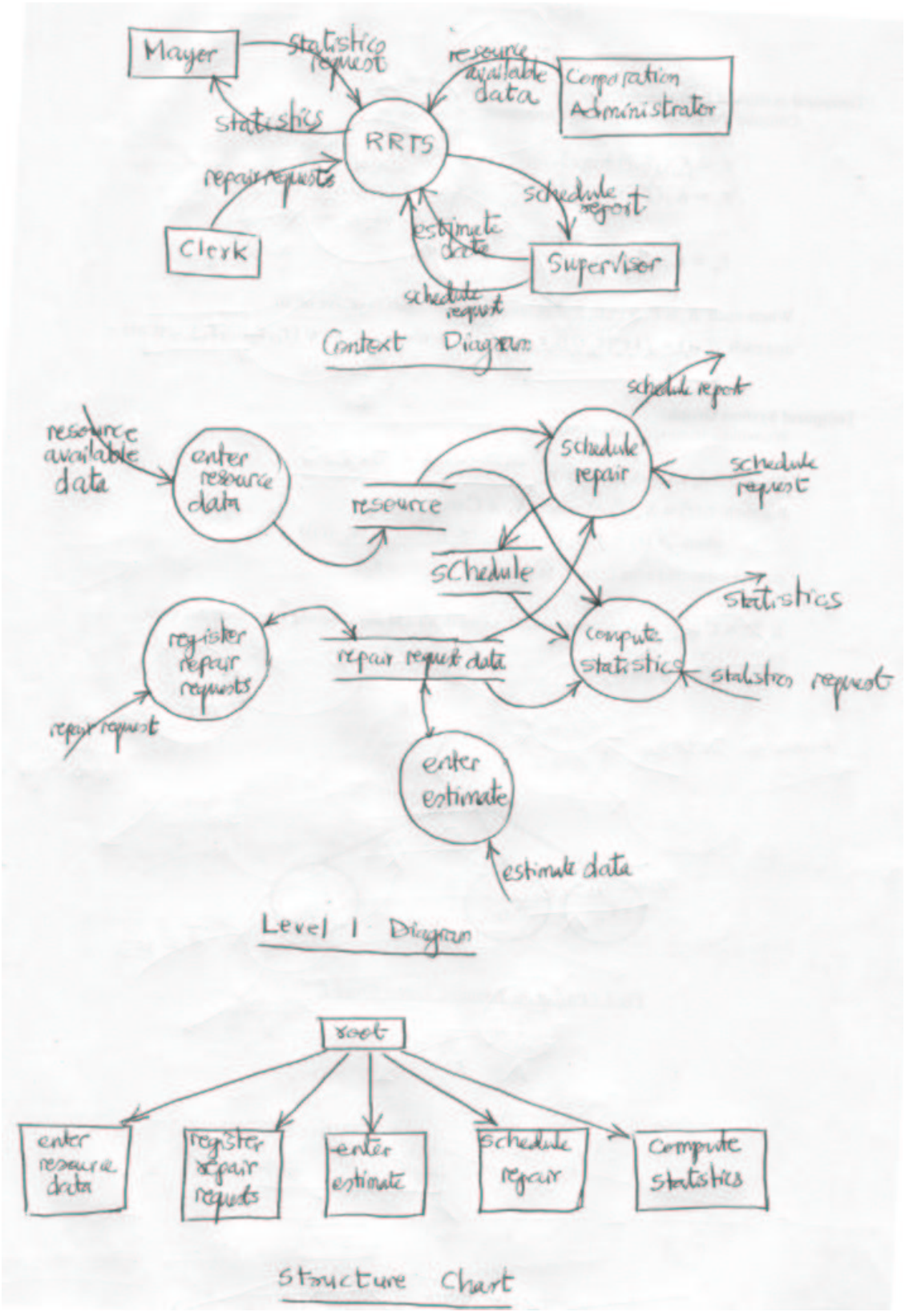

Figure 6.4: Structured Analysis and Design for Road Repair and Tracking System

this ingredient for the past three days and assuming that a minimum of two days stock must be maintained for all ingredients. Whenever the ordered ingredients arrive, the invoice data regarding the quantity and price is entered. If sufficient cash balance is available, the computer should print cheques immediately against invoice. Monthly sales receipt and expenses data should be generated whenever the manger would request to see them.

Ans:
See Fig. 5.

15. Perform structured analysis and design for the following Judiciary Information System (JIS) software.
The attorney general's office has requested us to develop a Judiciary Information System (JIS), to help handle court cases and also to make the past court cases easily accessible to the lawyers and judges.. For each court case, the name of the defendant, defendant's address, the crime type (e.g., theft, arson, etc.), when committed (date), where committed (location), name of the arresting officer, and the date of the arrest are entered by the court registrar. Each court case is identified by a unique case identification number (CIN) which is generated by the computer. The registrar assigns a date of hearing for each case. For this the registrar expects the computer to display the vacant slots on any working day during which the case can be scheduled. Each time a case is adjourned, the reason for adjournment is entered by the registrar and he assigns a new hearing date. If hearing takes place on any day for a case, the registrar enters the summary of the court proceedings and assigns a new hearing date. Also, on completion of a court case, the summary of the judgment is recorded and the case is closed but the details of the case is maintained for future reference. Other data maintained about a case include the name of the presiding judge, the public prosecutor, the starting date, and the expected completion date of a trial. The judges should be able to browse through the old cases for guidance on their judgment. The lawyers should also be permitted to browse old cases, but should be charged for each old case they browse. Using the JIS software, the Registrar of the court should be able to query the following:

   (a) The currently pending court cases.
   In response to this query, the computer should print out the pending cases sorted by CIN. For each pending case, the following data should be listed: the date in which the case started, the defendant's name, address, crime details, the lawyer's name, the

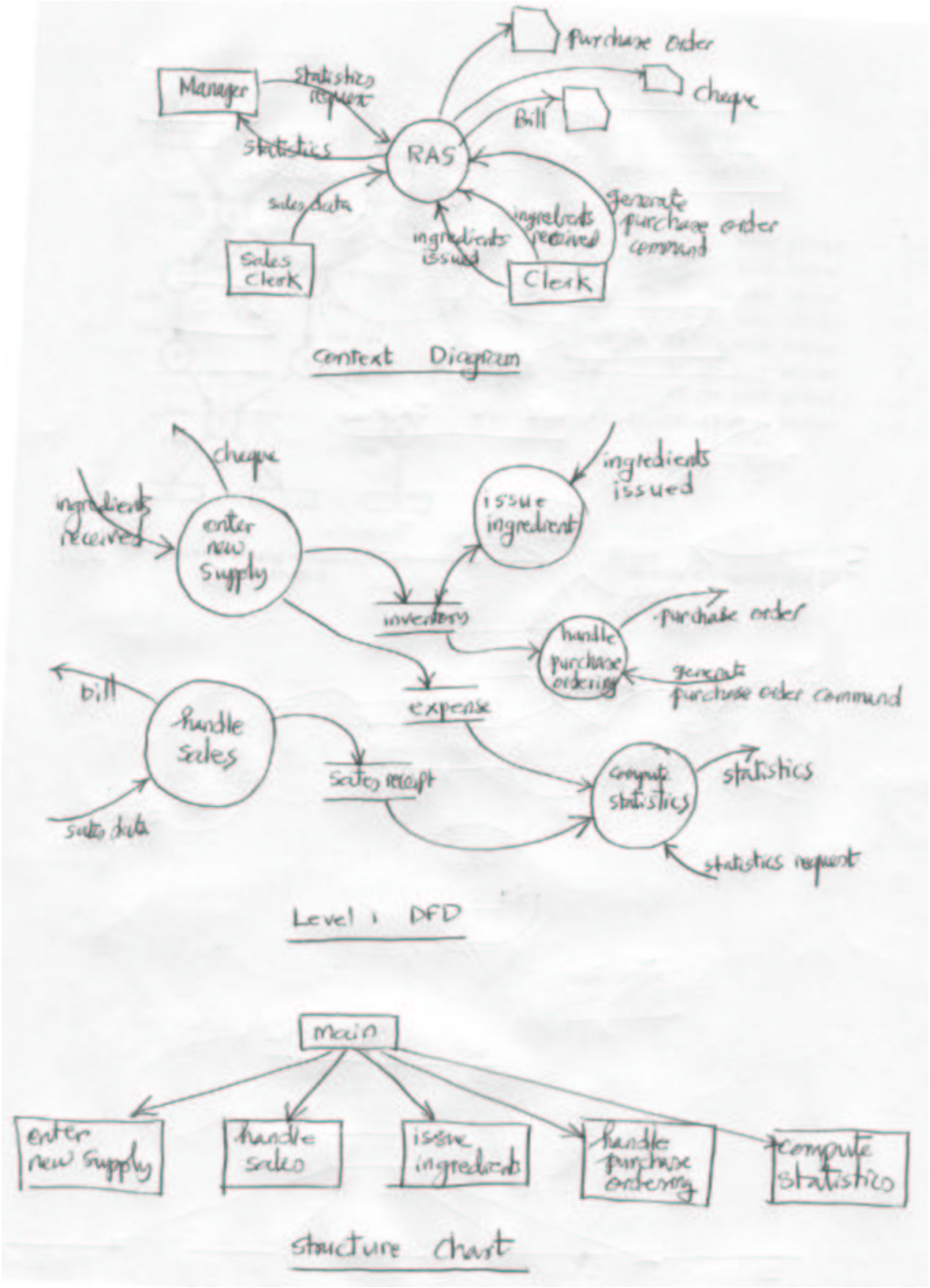

Figure 6.5: Structured Analysis and Design for Restaurant Automation Software

public prosecutor's name, and the attending judge's name.

(b) The cases that have been resolved over any given period.
The output in this case should chronologically list the starting date of the case, the CIN, the date on which the judgment was delivered, the name of the attending judge, and the judgment summary.

(c) The cases that are coming up for hearing on a particular date.

(d) The status of any particular case (cases are identified by CIN).

Ans:
See Fig. 7.

16. Perform structured analysis and structured design for the following Library Information System (LIS) software:

Library Information System (LIS): Different activities of the library of our institute pertaining to the issue and return of the books by the members of the library and various queries regarding books as listed below are to be automated.

- The library has 10,000 books. Each book is assigned a unique identification number (called ISBN number). The Library clerk should be able to enter the details of the book into the LIS through a suitable interface.
- There are four categories of members of the library: undergraduate students, post graduate students, research scholars, and faculty members.
- Each library member is assigned a unique library membership code number.
- Each undergraduate student can issue up to 2 books for 1 month duration.
- Each postgraduate student can issue up to 4 books for 1 month duration.
- Each research scholar can issue up to 6 books for 3 months duration.
- Each faculty member can issue up to 10 books for six months duration.
- The LIS should answer user queries regarding whether a particular book is available. If a book is available, LIS should display the rack number in which the book is available and the number of copies available.

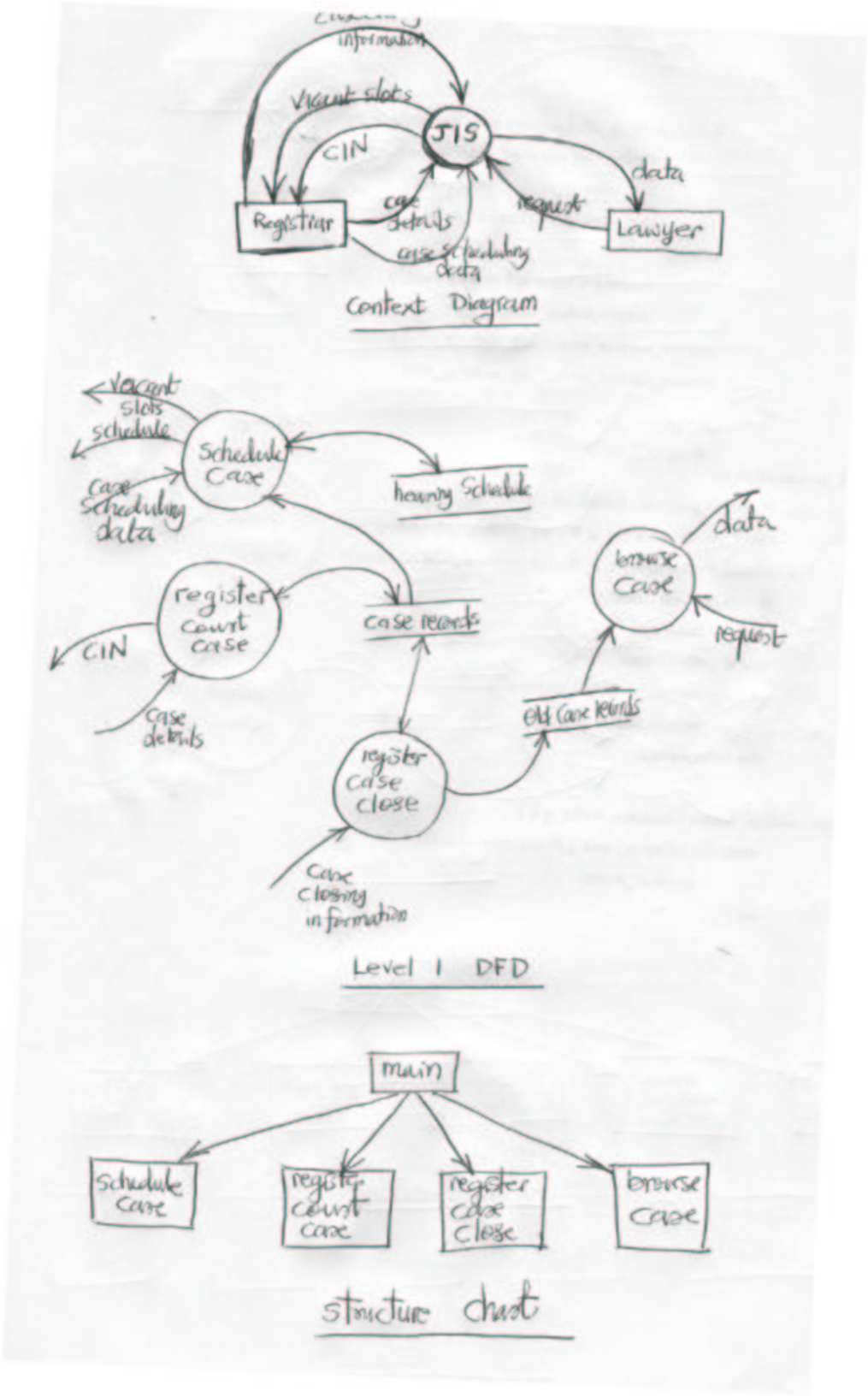

Figure 6.6: Structured Analysis and Design for Judiciary Automation Software

- LIS registers each book issued to a member. When a member returns a book, LIS deletes the book from the member's account and makes the book available for future issue.
- Members should be allowed to reserve books which have been issued out. When such a reserved book is returned, LIS should print a slip for the concerned member to get the book issued and should disallow issue of the book to any other member for a period of seven days or until the member who has reserved the books gets it issued.
- When a member returns a book, LIS prints a bill for the penalty charge for overdue books. LIS calculates the penalty charge by multiplying the number of days the book is overdue by the penalty rate.
- LIS prints reminder messages for the members against whom books are over due, upon a request by the Librarian.
- LIS should allow the Librarian to create and delete member records. Each member should be allocated a unique membership identification number which the member can use to issue, return, and reserve books.

Ans:
See Fig. 7.

17. Perform the SA/SD for the following word processing software.
Word processing software:

- The word processing software should be able to read text from an ASCII file or HTML file and store the formatted text as HTML files in the disk.
- The word processing software should ask the user about the number of characters in an output line of the formatted text. The user should be allowed to select any number between 1 and 132.
- The word processing software should process the input text in the following way.
  - Each output line is to contain exactly the number of characters specified by the user (including blanks).
  - The word processing software is to both left and right justify the text so that there are no blanks at the left- and right-hand ends of lines except the first and possibly the last lines of paragraphs. The word processing software should do this by inserting extra blanks between words.

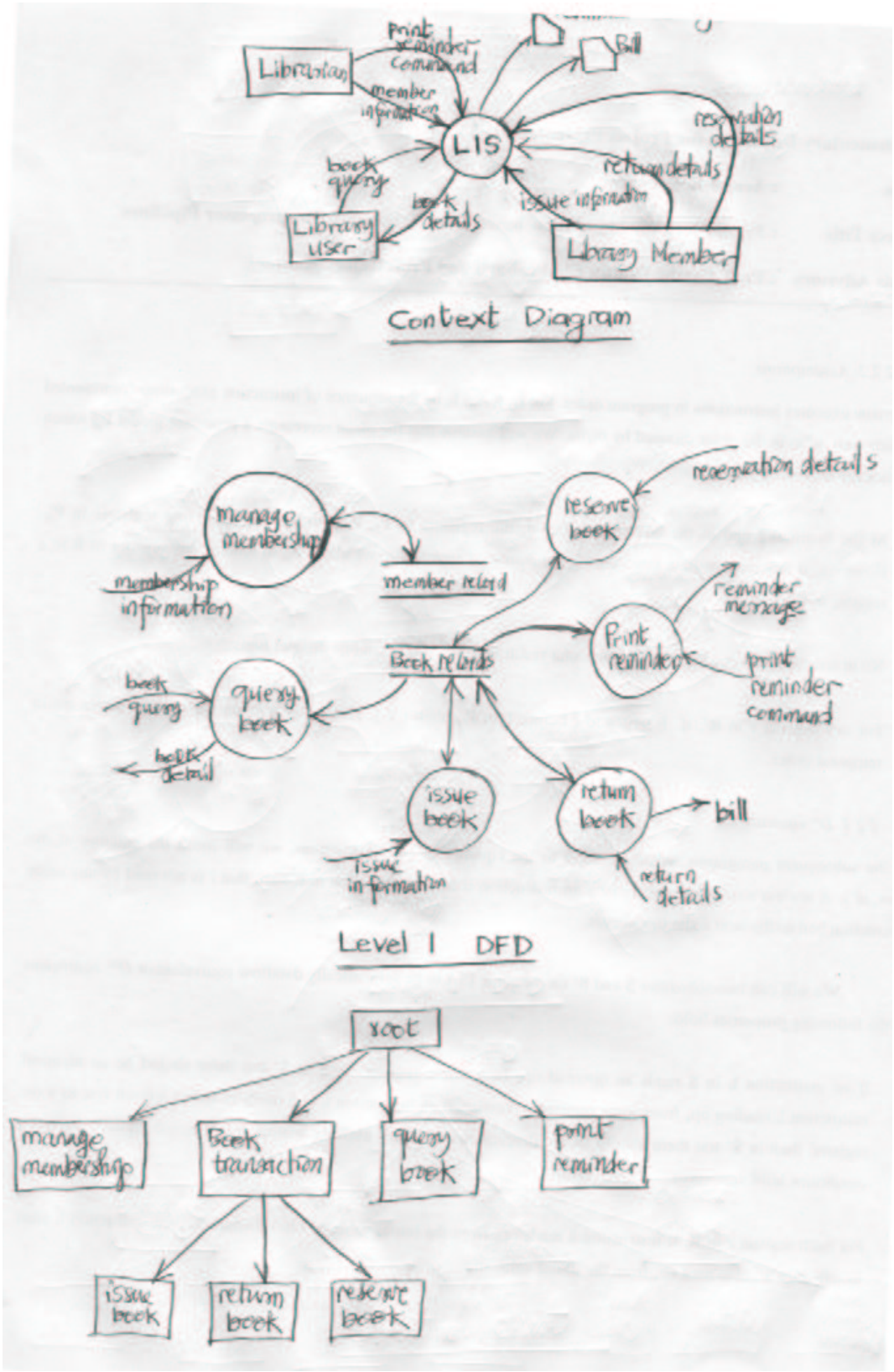

Figure 6.7: Structured Analysis and Design for Library Automation Software

  - The input text from the ASCII file should consist of words separated by one or more blanks and a special character PP, which denotes the end of a paragraph and the beginning of another.
  - The first line of each paragraph should be indented by five spaces and should be right justified.
  - The last line of each paragraph should be left justified.
- The user should be able to browse through the document and add, modify or delete words. He/she should also be able to mark any word as bold, italic, superscript, or subscript.
- The user can request to see the number of characters, words, lines, and paragraphs used in the document.
- The user should be able to save his documents under a name specified by him.

18. It is required to develop a graphics editor software package using which one can create/modify several types of graphics entities. In summary, the graphics editor should support the following features: (Those who are not familiar with any graphics editor, please look at the Graphics Drawing features available in either MS-Word or Powerpoint software. You can also examine any other Graphical Drawing package accessible to you. An understanding of the standard features of a Graphics Editor will help you understand the different features required.)

    - The graphics editor should support creating several types of geometric objects such as circles, ellipses, rectangles, lines, text, and polygons.
    - Any created object can be *selected* by clicking a mouse button on the object. A selected object should be shown in a highlighted color.
    - A selected object can be edited, i.e. its associated characteristics such as its geometric shape, location, color, fill style, line width, line style, etc. can be changed. For texts, the text content can be changed.
    - A selected object can be copied, moved, or deleted.
    - The graphics editor should allow the user to save his created drawings on the disk under a name he would specify. The graphics editor should also support loading previously created drawings from the disk.

- The user should be able to define any rectangular area on the screen to be *zoomed* to fill the entire screen.
- A *fit screen function* makes the entire drawing fit the screen by automatically adjusting the zoom and pan values.
- A pan function should allow the displayed drawing to be panned along any direction by a specified amount.
- The graphics editor should support grouping. A group is simply a set of drawing objects including other groups which when grouped behave as a single entity. This feature is especially useful when you wish to manipulate several entities in the same way. A drawing object can be a direct member of at most one group. It should be possible to perform several editing operations on a group such as move, delete, and copy.
- A set of 10 clip boards should be provided to which one can copy various types of selected entities (including groups) for future use in pasting these at different places when required.

19. Perform SA/SD for the following **Software component cataloguing software.**

    Software component cataloguing software: The software component cataloguing software consists of a software components catalogue and various functions defined on this components catalogue. The software components catalogue should hold details of the components which are potentially reusable. The reusable components can be either design or code. The design might have been constructed using different design notations such as UML,ERD,structured design, etc. Similarly, the code might have been written using different programming languages. A cataloguer may enter components in the catalogue, may delete components from the catalogue, and may associate reuse information with a catalogue component in the form of a set of key words. A user of the catalogue may query about the availability of a component using certain key words to describe the component. In order to help manage the component catalogue (i.e., periodically purge the unused components) the cataloguing software should maintain information such as how many times a component has been used, and how many times the component has come up in a query but not used. Since the number of components usually tend to be very high, it is desirable to be able to classify the different types of components hierarchically. A user should be able to browse the components in each category.

20. Perform structured analysis and structured design for developing the following Supermarket Automation software:
Supermarket Automation Software (SAS): The manager of a supermarket wants us to develop an automation software. The supermarket stocks a set of items. Customers pick up their desired items from the different counters in required quantities. The customers present these items to the sales clerk. The sales clerk enters the code number of these items along with the respective quantity/units.

    - SAS should at the end of a sales transaction print the bill containing the serial number of the sales transaction, the name of the item, code number, quantity, unit price, and item price. The bill should indicate the total amount payable.
    - SAS should maintain the inventory of the various items of the supermarket. The manager upon query should be able to see the inventory details. In order to support inventory management, the inventory of an item should be decreased whenever an item is sold. SAS should also support an option by which an employee can update the inventory when new supply arrives.
    - SAS should support printing the sales statistics for every item the supermarket deals with for any particular day or any particular period. The sales statistics should indicate the quantity of an item sold, the price realized, and the profit.
    - The manager of the supermarket should be able to change the the price at which an item is sold as the prices of the different items vary on a day-to-day basis.

21. Perform structured analysis and structured design for developing the following Transport Company Computerization (TCC) software:
Transport company computerization (TCC) software: A transport company wishes to computerize various book keeping activities associated with its operations.

    - A transport company owns a number of trucks.
    - The transport company has its head office located at the capital and has branch offices at several other cities.
    - The transport company receives consignments of various sizes at (measured in cubic meters) its different offices to be forwarded to different branch offices across the country.
    - Once the consignment arrives at the office of the transport company, the details of the volume, destination address, sender address, etc. are entered into the computer. The computer would

compute the transport charge depending upon the volume of the consignment and its destination and would issue a bill for the consignment.

- Once the volume of any particular destination becomes 500 cubic meters, the computerization system should automatically allot the next available truck.
- A truck stays with the branch office until the branch office has enough cargo to load the truck fully.
- The manager should be able to view the status of different trucks at any time.
- The manager should be able to view truck usage over a given period of time.
- When a truck is available and the required consignment is available for dispatch, the computer system should print the details of the consignment number, volume, sender's name and address, and the receiver's name and address to be forwarded along with the truck.
- The manager of the transport company can query the status of any particular consignment and the details of volume of consignments handled to any particular destination and the corresponding revenue generated.
- The manager should also be able to view the average waiting period for different consignments. This statistics is important for him since he normally orders new trucks when the average waiting period for consignments becomes high due to non-availability of trucks. Also, the manager would like to see the average idle time of the truck in the branch for a given period for future planning.

22. Draw level 0 (context level) and level 1 data flow diagram for the following students' academic record management software.

    Students' academic record management software.

    - A set of courses are created. Each course consists of a unique course number, number of credits, and the syllabus.
    - Students are admitted to courses. Each students' details include his roll number, address, semester number and the courses registered for the semester.
    - The marks of student for various units he credited are keyed in.

- Once the marks are keyed in, the SWA (semester weighted average) is calculated.
- The recent marks of the student are added to his previous marks and a weighted average based on the credit points for various units is calculated.
- The marks for the current semester are formatted and printed.
- The SWA appears on the report.
- A check must be made to determine if a student should be placed on the Vice-Chancellor's list. This is determined based on whether a student scores an SWA of 85 or higher.
- If the SWA is lower than 50, the student is placed on a conditional standing.

23. Perform structured analysis and structured design (SA/SD) for a software to be developed for automating various activities associated with developing a CASE tool for structured software analysis.

CASE tool for Structured Analysis:

- The case tool should support a graphical interface and the following features.
- The user should be able to draw bubbles, data stores, and entities and connect them using data flow arrows. The data flow arrows are annotated by the corresponding data names.
- Should support editing the data flow diagram.
- Should be able to create the diagram hierarchically.
- The user should be able to determine balancing errors whenever required.
- The software should be able to create the data dictionary automatically.
- Should support printing the diagram on a variety of printers.
- Should support querying the data items and function names. The diagrams matching the query should be shown.

24. Perform structured analysis and structured design (SA/SD) for a software to be developed for automating various activities associated with developing a CASE tool for structured software design. The summary

of the requirements are the following:

CASE tool for Structured Design:

- The case tool should support a graphical interface and the following features.
- It should be possible to import the DFD model developed by another program. The user should be able to apply the transform and transaction analysis to the imported DFD model.
- The user should be able to draw modules, control arrows, and data flow arrows. Also symbol for library modules should be provided. The data flow arrows are annotated with the corresponding data name.
- The modules should be organized in hierarchical levels.
- The user should be able to modify his design. Please note that when he deletes a data flow arrow, its annotated data name automatically gets deleted.
- For large software, modules may be hierarchically organized and clicking on a module should be able to show its internal organization.
- The user should be able to save his design and also be able to load previously created designs.

25. The local newspaper and magazine delivery agency has asked us to develop a software for him to automate various clerical activities associated with his business.

Newspaper Agency Automation Software.

- This software is to be used by the manager of the news agency and his delivery persons.
- For each delivery person, the system must print each day the publications to be delivered to each address.
- The customers usually subscribe one or more news papers and magazines. They are allowed to change their subscription notice by giving one week's advance notice.
- For each delivery person, the system must print each day the publications to be delivered to each address.

- The system should also print for the news agent the information regarding who received what and a summary information of the current month.
- At the beginning of every month bills are printed by the system to be delivered to the customers. These bills should be computed by the system automatically.
- The customers may ask for stopping the deliveries to them for certain periods when they go out of station.
- Customers may request to subscribe new newspapers/ magazines, modify their subscription list, or stop their subscription altogether.
- Customers usually pay their monthly dues either by cheque or cash. Once the cheque number or cash received is entered in the system, receipt for the customer should be printed.
- If any customer has any outstanding due for one month, a polite reminder message is printed for him and his subscription is discontinued if his dues remain outstanding for periods of more than two months.

26. Perform SA/SD for the following **University Department Information System.** This software concerns automating the activities of Department offices of Universities. Department offices in different universities do a lot of book-keeping activities the software to be developed targets to automate these activities.

    University Department Information System.

    - Various details regarding each student such as his name, address, course registered, etc. are entered at the time he takes admission.
    - At the beginning of every semester, students do course registration. The information system should allow the department secretary to enter data regarding student course registrations. As the secretary enters the roll number of each student, the computer system should bring up a form for the corresponding student and should keep track of courses he has already completed and the courses he has back-log, etc.
    - At the end of the semester, the instructors leave their grading information at the office which the secretary enter in the computer. The information system should be able to compute the

grade point average for each student for the semester and his cumulative grade point average (CGPA) and print the grade sheet for each student.

- The information system also keep s track of a inventories of the Department, such as equipments, their location, furnitures, etc.
- The Department has an yearly grant and the Department spends it in buying equipments, books, stationery items, etc. Also, in addition to the annual grant that the Department gets from the University, it gets funds from different consultancy service it provides to different organizations. It is necessary that the Department information system keeps track of the Department accounts.
- The information system should also keep track of the research projects of the Department, publications by the faculties, etc.

27. Perform SA/SD to develop a software to automate the activities of a small automobile spare parts shop. The small automobile spare parts shop sells the spare parts for a vehicles of several makes and models. Also, each spare part is typically manufactured by several small industries. To stream line the sales and supply ordering, the shop owner has asked us to develop the following motor part shop software. Perform the SA/SD for the **Motor Part Shop Software (MPSS).**

    Motor Part Shop Software (MPSS). The motor parts shop deals with large number of motor parts of various manufacturers and various vehicle types. Some of the motor parts are very small and some are of reasonably large size. The shop owner maintains different parts in wall mounted and numbered racks.

    The shop owner maintains as few inventory for each item as reasonable, to reduce inventory overheads after being inspired by the just-in-time (JIT) philosophy.

    Thus, one important problem the shop owner faces is to be able to order items as soon as the number of items in the inventory reduces below a threshold value. The shop owner wants to maintain parts to be able to sustain selling for about one week. To calculate the threshold value for each item, the software must be able to calculate the average number of parts sales for one week for each part.

At the end of each day, the shop owner would request the computer to generate the items to be ordered. The computer should print out the part number, the amount required and the address of the vendor supplying the part.

The computer should also generate the revenue for each day and at the end of the month, the computer should generate a graph showing the sales for each day of the month.

28. The students' society of a large college wishes to develop the following software to more efficiently manage the various shows conducted in their auditorium than the present manual system.

Students' Auditorium Management Software. Various types of social and cultural events are conducted in the students' auditorium. There are two categories of seats: balcony seats and ordinary seats. Normally balcony seats are more expensive in any show. The show manager fixes the price of these two categories of seats depending on the popularity of a show. The show manager also determines the number of balcony and ordinary seats that can be put on sale, since for each show some seats are offered as complimentary gifts to different functionaries of the students' society and to VIPs. The show manager also enters the show dates, the number of shows on any particular date and the show timings.

The spectators book their seats in advance by paying the full ticket price to the authorizes sales persons. The spectators indicate the type of the seat and the computer should print out the ticket clearly showing the seat numbers. The spectators' can cancel their booking before 3 clear days of the show. In this case the ticket price is refunded to them after deducting Rs.5/- as the booking charge per ticket. If a ticket is returned within 3 days and 1 day of a show, a booking charge of Rs.10/- is deducted for ordinary tickets and Rs.15/- is deducted for balcony tickets. On the last day of the show, there is a 50% deduction. The system should let the spectators query the availability of different classes of seats.

The show manager can query any time about the percentage of seats booked for various classes of seats and the amount collected in each case. The show manager creates login accounts for authorized sales

persons. When any authorizes sales person logs in and makes a sale, the computer should record the sales person's id in the sales transaction. This information would help in computing the commission payable to each sales person and also the amount collected by each sales person. These data can be queried by the show manager.

29. Perform SA/SD for the following problem. A transport company requires to automate its various operations. The company has a fleet of vehicles. Currently the company has the following vehicles :

    Ambassadors : 10 Non-AC, 2 AC
    Tata Sumo : 5 Non-AC, 5 AC
    Maruti Omni : 10 Non-AC
    Maruti Esteem : 10 AC
    Mahindra Armada : 10 Non-AC

    The company rents out vehicles to customers. When a customer requests for a car, the company lets them know what types of vehicles are available, and the charges for each car. For every car, there is a per hour charge, and a per kilometer charge. A car can be rented for a minimum of 4 hours. The amount chargeable to a customer is the maximum of (per hour charge for the car times the number of hours used, and per kilometer charge times the number of kilometers run) subject to a minimum amount decided by the charge for 4 hours use of the car. An AC vehicle of a particular category is charged 50% more than a non-AC vehicle of the same category. There is a charge of Rs 150 for every night halt regardless of the type of the vehicle.

    When a customer books a car, he has to deposit an advance amount. The customer also informs the company when he expects to return the car. When the car is returned, depending on the usage, either the customer is refunded some amount, or he has to pay additional amount to cover the cost incurred.

    The company can acquire new vehicles and add them to the fleet of its vehicles. Cars may be condemned and sold off. A car which is currently with the company can be in one of these three states: it may have gone for repair, it may be available, it may be rented out. If it is rented out, the company records the data and time when it has been rented out, and the mile-meter reading of the car at that time. The company also wants to maintain the amount of maintenance expense each vehicle incurs.

The company wants to collect statistics about various types of vehicles : the price of the car, average amount of money spent on repairs for the car, average demand, revenue earned by renting out the car, and fuel consumption of the car. Based on these statistics, the company may take a decision about which vehicles are more profitable. The statistics can also be used to decide the charge for different types of vehicles.

30. Perform structured analysis and structured design for the following *Medicine Shop Automation (MSA)* software:

    A retail medicine shop deals with a large number of medicines procured from various manufacturers. The shop owner maintains different medicines in wall mounted and numbered racks.

    - The shop owner maintains as few inventory for each item as reasonable, to reduce inventory overheads after being inspired by the just-in-time (JIT) philosophy.
    - Thus, one important problem the shop owner faces is to be able to order items as soon as the number of items in the inventory reduces below a threshold value. The shop owner wants to maintain medicines to be able to sustain selling for about one week. To calculate the threshold value for each item, the software must be able to calculate the average number of medicines sales for one week for each part.
    - At the end of each day, the shop owner would request the computer to generate the items to be ordered. The computer should print out the medicine description, the quantity required, and the address of the vendor supplying the medicine. The shop owner should be able to store the name, address, and the code numbers of the medicines that each vendor deals with.
    - Whenever new supply arrives, the shop owner would enter the item code number, quantity, batch number, expiry date, and the vendor number. The software should print out a cheque favoring the vendor for the items supplied.
    - When the shop owner procures new medicines it had not dealt with earlier, he should be able to enter the details of the medicine such as the medicine trade name, generic name, vendors who can supply this medicine, unit selling and purchasing price. The computer should generate a code number for this medicine which the shop owner would paste the code number in the rack where

this medicine would be stored. The shop owner should be able to query about a medicine either using its generic name or the trade name and the software should display its code number and the quantity present.

- At the end of every day the shop owner would give a command to generate the list of medicines which have expired. It should also prepare a vendor-wise list of the expired items so that the shop owner can ask the vendor to replace these items. Currently, this activity alone takes a tremendous amount of labour on the part of the shop owner and is a major motivator for the automation endeavor.
- Whenever any sales occurs, the shop owner would enter the code number of each medicine and the corresponding quantity sold. The MSA should print out the cash receipt.
- The computer should also generate the revenue and profit for any given period. It should also show vendor-wise payments for the period.

Ans:
See Fig. 8.

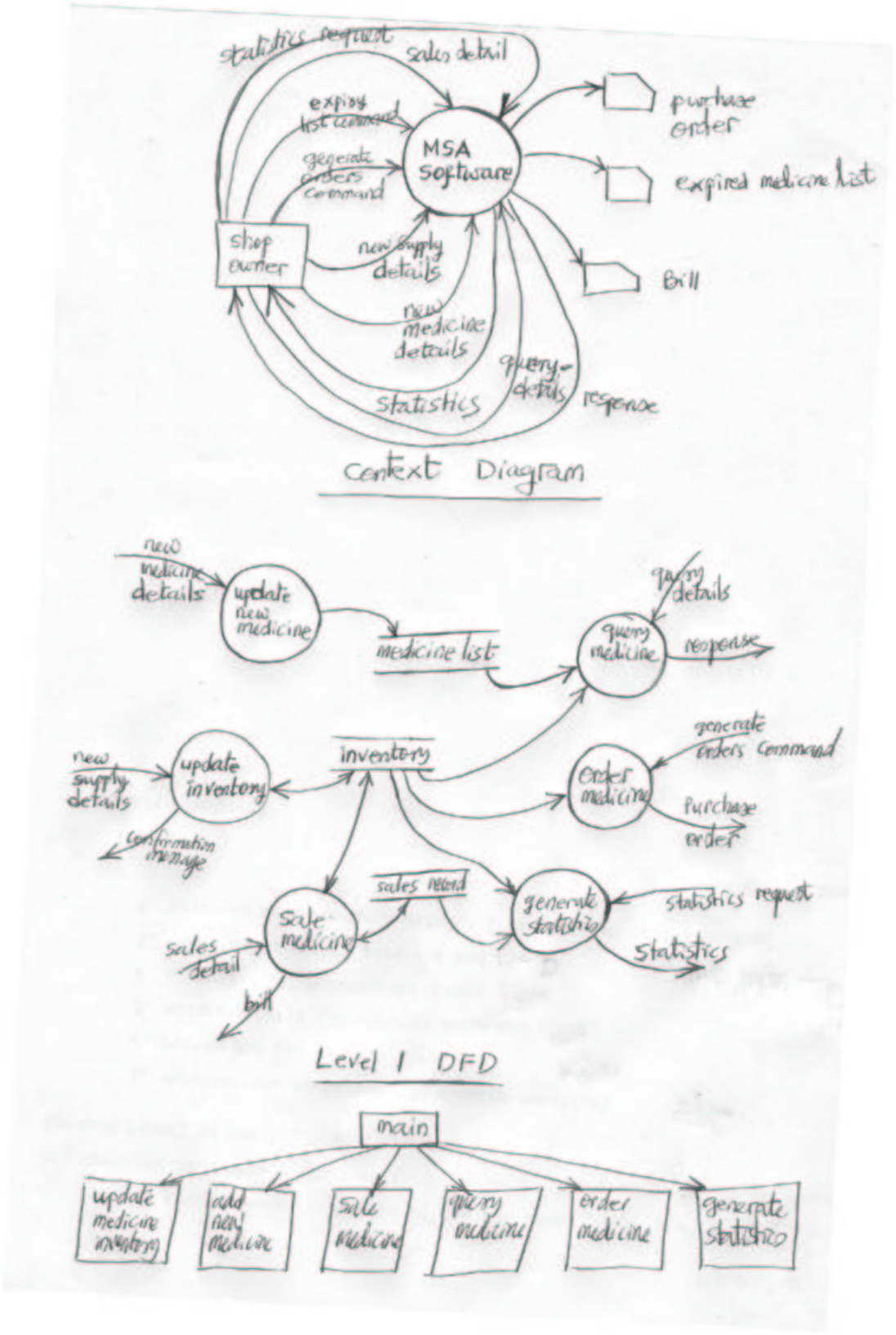

Figure 6.8: Structured Analysis and Design for Medicine shop Automation Software

Chapter 7

Object Modelling using UML

1. With the help of a suitable example explain how the inheritance feature of the object oriented paradigm helps in code reuse?
   Ans:
   The method and data defined in the base class are inherited (reused) by the derived classes.

2. With the help of a suitable example explain how *polymorphism* helps in developing easily maintainable and intuitively appealing code.
   Ans:
   Consider an object-oriented program and a traditional program for drawing various graphic objects on the screen. Assume that shape is the base class, and the classes circle, rectangle, and ellipse are derived from it. Now, shape can be assigned any objects of type circle, rectangle, ellipse, etc. But, a draw method invocation of the shape object would invoke the appropriate method.

   Suppose in the example program segment, it is later found necessary to handle a new graphics drawing primitive, say ellipse. Then, the procedural code has to be changed by adding a new `if-then-else` clause. However, in case of the object-oriented program, the code need not change, only a new class called ellipse has to be defined.

3. Explain the concept of dynamic binding as used in object-oriented languages using a simple illustrative example. How is dynamic binding useful?

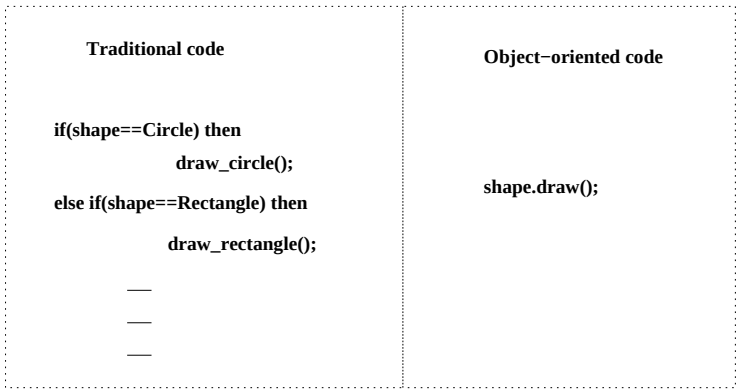

Figure 7.1: Traditional code and object-oriented code using dynamic bindin g

Ans:
Using dynamic binding a programmer can send a generic message to a set of objects which may be of different types (i.e., belonging to different classes) and leave the exact way in which the message would be handled to the receiving objects. Suppose we have a class hierarchy of different geometric objects in a drawing as shown in Fig. 7.1. Now, suppose the display method is declared in the shape class and is overridden in each derived class. If the different types of geometric objects making up a drawing are stored in an array of type shape, then a single a call to the display method for each object would take care to display the appropriate drawing element. dynamic binding is that it leads to elegant programming and facilitates code reuse and maintenance. With dynamic binding, new derived objects can be added with minimal changes to existing objects. We shall illustrate this advantage of polymorphism by comparing the code segments of an object-oriented program and a traditional program for drawing various graphic objects on the screen. Assume that shape is the base class, and the classes circle, rectangle, and ellipse are derived from it. Now, shape can be assigned any objects of type circle, rectangle, ellipse, etc. But, a draw method invocation of the shape object would invoke the appropriate method.

4. What are the reasons behind increased productivity when the object-oriented paradigm is adopted?
Ans:

Improved productivity comes about due to a variety of factors, such as:

- Code reuse by the use of predefined class libraries
- Code reuse due to inheritance
- Simpler and more intuitive abstraction, i.e. better organization of inherent complexity
- Better problem decomposition

5. What is the difference between method overloading and method overriding? Explain your answer by using a suitable example.
   Ans:
   When the same operation (e.g. create) is implemented by multiple methods, the method name is said to be overloaded. In a derived class the redefinition of methods which existed in the base class is called as *method overriding.*

6. Inheritance and composition (object embedding) can be considered to be similar in the sense that both require a copy of the component(base) to be embedded(linked) in the compound(derived) object. Is it possible to use object embedding (i.e. composite objects) to realize the features of inheritance and vice versa? Justify your answer with suitable examples.
   Ans:

7. What are the different system views that can be modelled using UML? What are the different UML diagrams which can be used to capture each of the views? Do you need to develop all the views of a system using all the modeling diagrams supported by UML? Justify your answer.
   Ans:
   The UML diagrams can capture the following views of a system:

   - User's view
   - Structural view
   - Behaviorial view
   - Implementation view
   - Environmental view

   User's view: use case diagram.

Structural view: Class diagram and object diagram.

Behaviorial view: Sequence diagram, collaboration diagram, activity diagram, state chart diagram.

Implementation view: Component diagram.

Environmental view: Deployment diagram.

No, we do not need to develop all the views of a system using all the modeling diagrams supported by UML.

8. State **TRUE** or **FALSE** of the following. Support your answer with proper reasoning:

   (a) An object oriented design must be implemented using an object-oriented language.
   Ans:
   FALSE.
   An object-oriented design can be implemented using a procedural language though it may require mosre effort to code the design compared to coding the design using an object-oriented language. In fact, initially C++ code was first preprocessed to C code before generating the machine code.

   (b) Any language directly supporting abstract data types (ADTs) can be called as an object-oriented language.
   Ans:
   False.
   It cannot be called object-oriented unless it supports inheritance.

   (c) In contrast to an abstract class, a concrete class should define all its data and methods in the class definition itself without inheriting any of them.
   Ans:
   False.
   A class is called concrete, when it supports the required constructors. It can inherit any methods from the base classes.

   (d) An object-oriented language can be used to implement function-oriented designs.
   Ans:
   True.
   In the degenerate case, a single class can provide all the functionalities required.

   (e) The inheritance relationship describes the *has a* relationship among the classes.

Ans:
False.
The inheritance relationship describes the *is a* relationship among the classes

(f) Inheritance feature of the object oriented paradigm helps in code reuse.
Ans:
TRUE.
The method and data defined in the base class are inherited (reused) by the derived classes.

(g) Object embedding (i.e. composite objects) can be equally effectively be used to have the effect of the composite class being derived from the component classes in lieu of using the inheritance relationship.
Ans:
FALSE.
It is very difficult to support inheritance in a class hierarchy using this approach. Also, dynamic binding would become difficult to support.

(h) Aggregation relationship between classes is antisymmetric.
Ans:
TRUE.
While the aggregate class contains the component class, the component class does not contain the aggregate class.

(i) The aggregation relationship can be recursively defined, i.e an object can contain instances of itself.
Ans:
TRUE.
A class (e.g. paragraph) can contain itself.

(j) State chart diagrams in UML are normally used to model how some behavior of a system is realized through the co-operative actions of several objects.
Ans:
FALSE.
The state chart diagrams are used to model how the behavior of a single object changes with time.

(k) Multiple inheritance is the feature by which many subclasses can inherit features of one base class.
Ans:
FALSE.

Multiple inheritance is the feature by which a subclass inherits features from multiple base classes.

(l) Class diagrams developed using UML can serve as the functional specification of a system.
Ans:
FALSE.
Use case diagrams developed using UML can serve as the functional specification of a system.

(m) An important advantage of polymorphism is facilitation of reuse.
Ans:
FALSE.
Polymorphism facilitates maintenance.

(n) A static object-oriented model should capture attributes and methods of classes and how different methods invoke each other.
Ans:
FALSE.
Though a static object-oriented model should capture attributes and methods of classes, itshould not capture how and in what order the different methods invoke each other.

(o) In a UML class diagram, the aggregation relationship defines an equivalence relationship among objects.
Ans:
FALSE.
Aggregation is not reflexive and neither symmetric.

(p) Abstract classes and interface classes (as used in UML, Java, etc.) are equivalent concepts.
Ans:
FALSE.
Abstract classes can contain data definitions and method prototypes, but an interface class cannot contain any data definition and contains only method prototypes.

(q) The aggregation relationship can be considered to be a special type of association relationship.
Ans:
TRUE.
Aggregation automatically implies that the component class is associated with the aggregate class and vice versa.

(r) The aggregation relationship can be reflexive.
Ans:

FALSE.
An object cannot contain instances of itself.

(s) The aggregation relationship can not be reflexive and symmetric but is transitive.
Ans:
FALSE.
Any relation that is not reflexive and symmetric cannot be transitive.

(t) Normally, you use an interaction diagram to represent how the behavior of an object changes over its life time.
Ans:
FALSE.
An interaction diagram captures how different objects interact with each other to exhibit a behavior.

(u) There might be several methods in a class implementing the same operation.
Ans:
TRUE.
This can be due to polymorphism.

(v) The chronological order of the messages in an interaction diagram cannot be determined from an inspection of the diagram.
Ans:
FALSE.
It can be easily determined by reading the diagram from top to bottom.

(w) The interaction diagrams can be effectively used to describe how the behavior of an object changes across several use case.
Ans:
FALSE.
An interaction diagram captures how objects interact to implement a single use case.

(x) A state chart diagram is good at describing behavior that involves multiple objects cooperating with each other to achieve some behavior.
Ans:
FALSE.
State chart diagram captures how the behavior of a single object changes with time.

(y) An object-oriented program that does not use the inheritance mechanism in the class definitions, cannot display dynamic bind-

ing.
Ans:
TRUE.
Dynamic binding occurs only when we have a inheritance hierarchy, and an object is assigned to another object of its ancestor class.

(z) The terms method and operation are equivalent concepts and can be used interchangeably.
Ans:
FALSE.
Several methods can implement the same operation. There is a subtle difference between the two terms.

() The effort to test and debug an object-oriented program is reduced if the number of message exchanges among objects is decreased.
Ans:
TRUE.
When the number of message exchanges increase, then the behavior of several objects and methods must be analyzed to test and debug the program.

9. What do you understand by the term encapsulation in the context of software design? What are the advantages of encapsulation?
Ans:
The property of an object by which it interfaces with the outside world only through messages is referred to as *encapsulation*. The data of an object are encapsulated within its methods and are available only through message-based communication. Encapsulation offers three important advantages:

- It protects an object's variables from corruption by other objects. This protection includes protection from unauthorized access and protection from different types of problems that arise from concurrent access of data such as deadlock and inconsistent values.
- Encapsulation hides the internal structure of an object so that interaction with the object is simple and standardized. This facilitates reuse of objects across different projects. Furthermore, if the internal structure or procedures of an object are modified, other objects are not affected. This results in easy maintenance and bug correction.

- Since objects communicate among each other using messages only, they are weakly coupled. The fact that objects are inherently weakly coupled enhances understandability of design since each object can be studied and understood almost in isolation from other objects.

10. How does data abstraction help in reducing the coupling in a design solution?
    Ans:
    Data abstraction means that each object hides (abstract away) from other objects the exact way in which its internal information is organized and manipulated. It only provides a set of methods, which other objects can use for accessing and manipulating this private information of the object. Thus, the coupling is lower compared to the case where the data can be directly modified.

11. Abstract classes cannot have instances. What is then the use of defining abstract classes?
    Ans:
    Abstract classes help push reusable methods up in the class hierachy thus increasing reuse, enhancing undersatandability, and reducing the overall system development effort.

12. (a) Point out the main differences between an object-oriented language (e.g. C++) and a procedural language (e.g. C).
    Ans:
    An object-oriented language such as C++ supports the basic object- orientation features such as class, inheritance, polymorphism, etc. besides supporting the basic constructs of a procedural language.

    (b) Can an object-oriented design be implemented using a procedural language? Can a traditional function-oriented design be implemented using an object-oriented language? Write the reason behind your answer.
    Ans:
    Yes, a procedural language can be used to implement an object-oriented design. The basic features of an object-oriented language such as class, inheritance, polymorphism can be supported through additional code. In fact, initially C++ programs were generating C code after the preprocessing step of compilation.

    Similarly a procedural design can be trivially coded in an object-oriented language as the data and methods of a single class.

13. What basic features a programming language needs to support in order to be called as an object oriented language? How is an object oriented programming language different from a traditional procedural programming language such as C or PASCAL?
Ans:
An object-oriented language such as C++ supports the basic object-orientation features such as class, inheritance, polymorphism, etc. besides supporting the basic constructs of a procedural language.

14. What is the difference between a sequence diagram and a collaboration diagram? In what context would you use each?
Ans:
A collaboration diagram shows both structural and behaviorial aspects explicitly. This is unlike a sequence diagram which shows only the behaviorial aspects. The structural aspect of a collaboration diagram consists of objects and the links existing between them. In this diagram, an object is also called a collaborator. The behaviorial aspect is described by the set of messages exchanged among the different collaborators.

    Sequence diagrams are used to determine the methods that each class should support for implementing various use cases. A use of the collaboration diagrams in our development process would be to help us to determine which classes are associated with which other classes.

15. What is a *stereotype* in UML? Explain with some example situations where these can be used?
Ans:
One of the main objectives of the creators of the UML was to restrict the number of primitive symbols in the language. A language having a large number of primitive symbols not only becomes difficult to learn but also makes it difficult to use these symbols and understand a diagram constructed by using a large number of basic symbols. The creators of UML introduced a *stereotype* which when used to annotate a basic symbol can slightly change the meaning of a basic symbol. Just as you stereotype your friends in the class as studious, jovial, serious, etc. stereotyping can be used to give special meaning to any basic UML construct.

    Both the human users and the external systems can be represented by stick person icons. When a stick person icon represents an external system, it is annotated by the stereotype ≪external system≫.

16. How can you specify different constraints on the modelling elements in UML? For example, how can you specify that all books are kept alphabetically sorted in a library?
Ans:
UML allows you to use any free form expression within curly brackets to describe constraints. The constraint {alphabetically sorted} associated with book class describes that books are kept alphabetically sorted.

17. What is the difference between static and dynamic models in the context of object-oriented modelling of systems? Identify the UML diagrams which provide these two models respectively.
Ans:
A static model captures the time invariant part of a system, whereas a dynamic model captures the time-dependent behavior of the system. The class and object diagrams describe the static structure of the system.

The dynamic model is captured by the sequence diagram, collaboration diagram, state chart diagram, and activity diagrams.

18. In the object-oriented modeling of systems using UML, how are the classification of users of a system into various types of actors and their representation in the use case diagram helpful in system development?
Ans:
One possible use of identifying the different types of users (actors) is in identifying and implementing a security mechanism through a login system, so that each actor can invoke only those functionalities to which he is entitled to. Another possible use is in preparing the documentation (e.g. users' manual) targeted at each category of user. Further, actors help in identifying the use cases and understanding the exact functioning of the system.

19. Draw a class diagram using the UML syntax to represent the fact that an order consists of one or more order items. Each order item contains the name of the item, its quantity and the date by which it is required. Each order item is described by an item order specification object having details such as its vendor addresses, its unit price, and manufacturer.
Ans:
See Fig. 2.

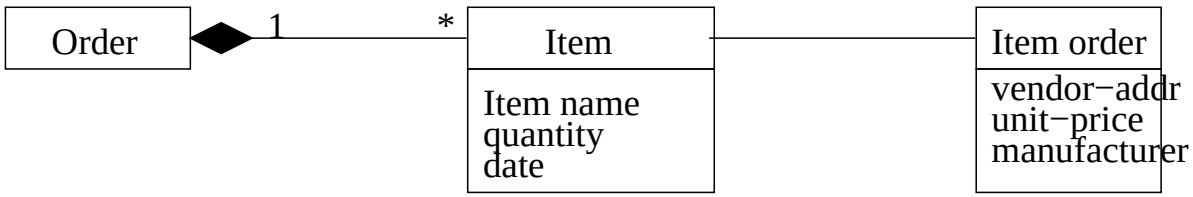

Figure 7.2: Class Diagram for Order

Chapter 8

Object-Oriented Software Development

1. What is the difference between Object Oriented Analysis (OOA) and Object Oriented Design (OOD)?
   Ans:
   The term object oriented analysis (OOA) refers to a method of developing an initial model of the software from the requirements specification. The analysis model is refined into a design model through Object Oriented Design (OOD).

2. Do you need to develop all the views of a system using all the modeling diagrams supported by UML? Justify your answer.
   Ans:
   The answer is "No". For a simple system, the use case model, class diagram, one of the interaction diagrams may be sufficient. For a system in which the objects undergo many state changes, a state chart diagram may be necessary. For a system, which is implemented on a large number of hardware components, a deployment diagram may be necessary. So, the type of models to be constructed depends on the problem at hand.

3. What are **design patterns**? What are the advantages of using design patterns? Name some popular design patterns.
   Ans:
   Patterns are standard solutions to some commonly recurring problems. A pattern serves as a guide for creating a "good" design. Patterns are based on sound common sense and the application of funda-

mental design principles. Once we become familiar with the patterns, we can spot them in our development projects and reuse them.

Some important patterns are the following:

- **Expert**
- **Creator**
- **Controller**
- **Facade**

4. State whether the following statements are **TRUE** or **FALSE**. Give reasons behind your answers.

   (a) Deep class hierarchies are signs of an object-oriented design done well.
   Ans:
   FALSE
   When the class hierarchies are deep, the classes at the bottom of the hierarchy tend to have too many methods —- making it very difficult to understand the behavior of these classses.

   (b) A large number of message exchanges among objects during the realization of a use case indicates effective delegation of responsibilities and is a sign of good design.
   Ans:
   FALSE
   A large number of message exchanges makes it difficult to understand the design and debug it.

5. Define the term *cohesion* in the context of object-oriented design of systems.
   Ans:

6. What are some of the important features that characterize a good object-oriented design?
   Ans:

7. Briefly outline the important steps involved in developing a software system using a popular Object-oriented design methodology.
   Ans:
   In OOD, we are concerned about cohesion at three levels:

- *Cohesiveness of the individual methods.* Cohesiveness of each of the individual method is desirable, since it assures that each method does only a well-defined function.
- *Cohesiveness of the data and methods within a class.* This is desirable since it assures that the methods of an object do actions for which the object is naturally responsible, i.e. it assures that no action has been improperly mapped to an object.
- *Cohesiveness of an entire class hierarchy.* Cohesiveness of methods within a class is desirable since it promotes encapsulation of the objects.

8. Perform the Object-Oriented Design for the development of the *Hotel automation software* of Chap. 5.

9. Perform the Object-Oriented Design for the development of the *Road Repair and Tracking Software (RRTS)* of Chap. 5.

10. Perform the Object-Oriented Design for the development of the *Book shop Automation Software (BAS)* of Chap. 5.

11. Perform the Object-Oriented Design for the development of the *Library Information System (LIS) software* of Chap. 5.

12. Perform object-oriented design for developing the following simulation software:
A factory has a certain category of machines that require frequent adjustments and repair. Each category of machine fails uniformly after continuous operation and the failure profile of the different categories of machines is given by its mean time to failure (MTTF). A certain number of adjusters are employed to keep the machines running. A service manager coordinates the activities of the adjusters. The service manager maintains a queue of inoperative machines. If there are machines waiting to be repaired, the service manager assigns the machine at the front of the queue to the next available adjuster. Likewise, when some adjusters are not busy, the service manager maintains a queue of idle adjusters and assigns the adjuster at the front of the queue to the next machine that breaks down.

At any given time, one of the two queues will be empty. Thus, the service manager needs to maintain only a single queue, which when it is not empty contains only machines or only adjusters. The factory management wishes to get as much as possible out of its machines and adjusters. It is therefore interested in machine utilization — the percentage of time a machine is up and running and the adjuster utilization — the percentage of time an adjuster is busy. The goal of our simulation is then to see how the average machine and adjuster utilization depend on such factors as the number of machines, the number of adjusters, the reliability of the machines in terms of mean time to failure (MTTF), and the productivity of the adjusters.

13. Consider the following Elevator Control Problem.

A software system (Elevator Controller) must control a set of 4 elevators for a building with 10 floors. Each elevator contains a set of buttons, each corresponding to a desired floor. These are called *floor request buttons*, since they indicate a request to go to a specific floor. Each elevator as well has a *current floor indicator* above the door. Each floor has two buttons for requesting elevators called *elevator request buttons* because they request an elevator. The Elevator Controller will receive all the signals from the passengers and decide on the control actions to be fed to the elevators

Each floor has a sliding door for each shaft arranged so that two door halves meet in the center when closed. When the elevator arrives at the floor, the doors opens at the same time the door on the elevator opens. The floor does have both pressure and optical sensors to prevent closing when an obstacle is between the two doors halves. If an obstruction is detected by either sensor, the door shall open. The door shall automatically close after a timeout period of five second after the door opens. The detection of an obstruction shall restart the door closure time after an obstruction is removed. There is a speaker on each floor that pings in response to the arrival of an elevator.

On each floor, there are two elevator request buttons, one for UP and one for DOWN. On each floor above each elevator door, there is an indicator specifying the current floor of the elevator and another indicator for its current direction. The system shall respond to an elevator request by sending the nearest elevator that is either idle or already going in the requested direction. If no elevators are currently

available, the request shall be pending until an elevator meets the previously mentioned criterion. Once pressed, the request buttons are backlit to indicate that a request is pending. Pressing an elevator request button when a request for that direction is already pending, shall have no effect. When an elevator arrives to handle the request (i.e., it is slated to go in the selected direction), then the door shall stop closing and the door closure timer shall be reset.

To enhance safety, a cable tension sensor monitors the tension on the cable controlling the elevator. In the event of a failure, in which the measured tension falls below a critical value, then four external locking clamps connected to running tracks in the shaft stop the elevator and hold it in place.

(a) Develop the domain model.

(b) Develop state chart model for the classes possessing significant number of states and behavior.

Chapter 9

User Interface Design

1. List five desirable characteristics that a good user interface should possess.
   Ans:
   In the following, we identify a few important characteristics of a good user interface:

   Speed of learning.
   Use of Metaphors and intuitive command names.
   Consistency.
   Component-based interface.
   Speed of use.
   Speed of recall.
   Error prevention.
   Attractiveness.
   Consistency.
   Feedback.
   Support for multiple skill levels.
   Error recovery (undo facility).
   User guidance and on-line help.

2. Compare the relative advantages of textual and graphical user interfaces.
   Ans:
   Let us compare various characteristics of a GUI with those of a text-based user interface.

- In a GUI multiple windows with different information can simultaneously be displayed on the user screen.
- Iconic information representation and symbolic information manipulation is possible in a GUI.
- A GUI usually supports command selection using an attractive and user-friendly menu selection system.
- In a GUI, a pointing device such as a mouse or a light pen can be used for issuing commands. The use of a pointing device increases the efficacy of command issue procedure.
- On the flip side, a GUI requires special terminals with graphics capabilities for running and also requires special input devices such a mouse.

3. What is the difference between user guidance and on-line help system in the user interface of a software system? Discuss the different ways in which on-line help can be provided to a user while he is executing the software.
   Ans:

4. (a) Compare the relative advantages of command language, menu-based, and direct manipulation interfaces.
   Ans:
   Among the three categories of interfaces, the command language interface allows for most efficient command issue procedure requiring minimal typing. However, command language-based interfaces are difficult to learn and require the user to memorize the set of primitive commands.

   An important advantage of a menu-based interface over a command language-based interface is that a menu-based interface does not require the users to remember the exact syntax of the commands. A menu-based interface is based on recognition of the command names, rather than recollection. Further, in a menu-based interface the typing effort is minimal as most interactions are carried out through menu selections using a pointing device.

   Important advantages of iconic interfaces include the fact that the icons can be recognized by the users very easily, and that icons are language-independent. However, direct manipulation interfaces can be considered slow for experienced users. Also, it is

difficult to give complex commands using a direct manipulation interface.

(b) Suppose you have been asked to design the user interface of a large software product. Would you choose a menu-based, a direct manipulation, a command language-based, or a mixture of all these types of interfaces to develop the interface for your product? Justify your choice.
Ans:
It would be appropriate to use a mixture of all these types of interfaces to develop the interface for the product. Users start as novices, but become more and more proficient as they continue to use the software. Thus, when all the three types of interfaces are available, the user can choose the interface that best suites him depending on his experience level.

5. State **TRUE** or **FALSE** of the following. Support your answer with proper reasoning:

(a) Visual programming style is restricted to user interface development only.
Ans:
FALSE.
Visual programming has been used successfully in many other areas such as CAD (Computer-Aided Design), animation applications, etc.

(b) Novice users normally prefer command language interfaces over both menu-based and iconic interfaces.
Ans:
FALSE.
Novice users prefer iconic interfaces, as the commands are self-evident and requires minimal typing.

(c) For modern user interfaces, LOC is an accurate measure of the size of the interface.
Ans:
FALSE.
Since modern user interfaces are developed using a user interface component paradigm, LOC is not a meaningful measuer of the size of an interface. A more accurate measure is widget-points.

6. Discuss why several popular software packages support a command language in addition to menu-based and iconic user interfaces.
Ans:

Users start as novices, but become more and more proficient as they continue to use the software. Thus, when all the three types of interfaces are available, the user can choose the interface that best suites him depending on his experience level.

7. List the important advantages and disadvantages of a command language interface.
   Ans:
   Among the three categories of interfaces, the command language interface allows for most efficient command issue procedure requiring minimal typing. However, command language-based interfaces are difficult to learn and require the user to memorize the set of primitive commands.

8. List the important advantages and disadvantages of a menu-based interface.
   Ans:
   An important advantage of a menu-based interface over a command language-based interface is that a menu-based interface does not require the users to remember the exact syntax of the commands. A menu-based interface is based on recognition of the command names, rather than recollection. Further, in a menu-based interface the typing effort is minimal as most interactions are carried out through menu selections using a pointing device.

9. Compare the relative advantages of scrolling menu, hierarchical menu, and walking menu as techniques for organizing user commands.
   Ans:
   A scrolling menu enables the user to view and select the menu items that can not be accommodated on the screen. However, in a scrolling menu all the commands should be highly correlated, so that the user can easily locate a command that he needs.

   Hierarchical menu can be used to manage large number of choices, but the users are likely to face navigational problems because they might lose track of where they are in the menu tree. This probably is the main reason why this type of interface is very rarely used.

   Important advantages of iconic interfaces include the fact that the icons can be recognized by the users very easily, and that icons are language-independent. However, direct manipulation interfaces can be considered slow for experienced users. Also, it is difficult to give complex commands using a direct manipulation interface.

10. What do you understand by an iconic interface? Explain how you can issue commands using an iconic interface.
Ans:
Iconic interfaces present the interface to the user in the form of visual models. For this reason, direct manipulation interfaces are sometimes called as *iconic interfaces.* In this type of interface, the user issues commands by performing actions on the visual representations of the objects, e.g. pull an icon representing a file into an icon representing a trash box, for deleting the file.

11. List the important advantages and disadvantages of a direct manipulation interface.
Ans:
Important advantages of iconic interfaces include the fact that the icons can be recognized by the users very easily, and that icons are language-independent. However, direct manipulation interfaces can be considered slow for experienced users. Also, it is difficult to give complex commands using a direct manipulation interface.

12. List the important GUI components using which you can develop a graphical user interface for any application.
Ans:
The following are some important GUI components using which you can develop a graphical user interface for any application.

- *Label*
- *Container*
- *Pop-up menu.*
- *Pull-down menu.*
- *Dialog boxes.*
- *Push button.*
- *Radio buttons.*
- *Combo boxes.*

13. What is the difference between a mode-based and a modeless user interface? What are their relative advantages? Which one would you use for developing the interface of a software product supporting a large number of commands? Justify your answer.
Ans:
A *mode* is a state or collection of states in which only a subset of all

user interaction tasks can be performed. In a modeless interface, the same set of commands can be invoked at any time during the running of the software. Thus, a modeless interface has only a single mode and all the commands are available all the time during the operation of the software. On the other hand, in a mode-based interface, different sets of commands can be invoked depending on the mode in which the system is, i.e. the mode at any instant is determined by the sequence of commands already issued by the user.

For developing the interface of a software product supporting a large number of commands, a mode-based interface would be suitable as otherwise if a large number of commands are supported using a modeless interface, the interface would be very difficult to learn and use.

14. What is a Window Management System (WMS)? Represent the main components of a WMS in a schematic diagram and briefly explain their roles.
Ans:
A window management system is primarily a resource manager. It keeps track of the screen area resource and allocates it to the different windows that seek to use the screen. It also provides the basic behavior to the windows and provides several utility routines to the application programmer for user interface development. A WMS simplifies the task of a GUI designer to a great extent by providing the basic behavior to the various windows such as move, resize, iconify, etc. as soon as they are created and by providing the basic routines to manipulate the windows from the application program such as creating, destroying, changing different attributes of the windows, and drawing text, lines, etc.

The schematic diagram of a WMS is shown in Fig. 1

15. What are the advantages of using a window management system for GUI design? Name some commercially available window management systems.
Ans:
A WMS simplifies the task of a GUI designer to a great extent by providing the basic behavior to the various windows such as move, resize, iconify, etc. as soon as they are created and by providing the basic routines to manipulate the windows from the application program such as creating, destroying, changing different attributes of the windows, and drawing text, lines, etc.

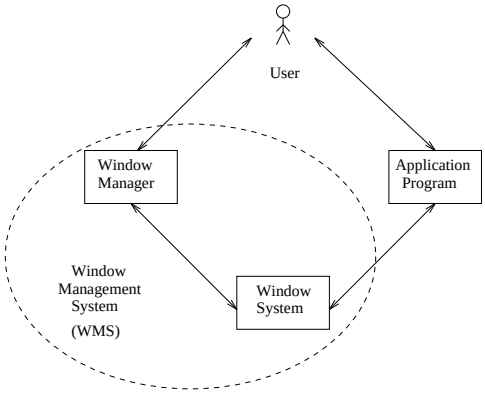

Figure 9.1: Window management system

Names of commercial WMS are: Motif, OWLM (OpenLook Window Manager), KWM, TWM, etc.

16. Briefly discuss the architecture of the X window system. What are the important advantages of using the X window system for developing graphical user interfaces?
Ans:
The X architecture is pictorially depicted in Fig. 2.

17. Write five sentences to highlight the important features of Visual C++?
Ans:
Visual C++ provides tools for building programs with window-based user interfaces for Microsoft Windows environments. In visual C++ you usually design menu bars, icons, and dialog boxes, etc. before adding them to your program. These objects are called as resources. You can design shape, location, type, and size of the dialog boxes before writing any C++ code for the application.

18. What do you understand by a *visual language*? How do languages such as Visual C++ and Visual Basic let you do component-based user interface development?
Ans:
Visual programming is the drag and drop style of program development. In this style of user interface development, a number of visual objects (icons) representing the GUI components are provided by the

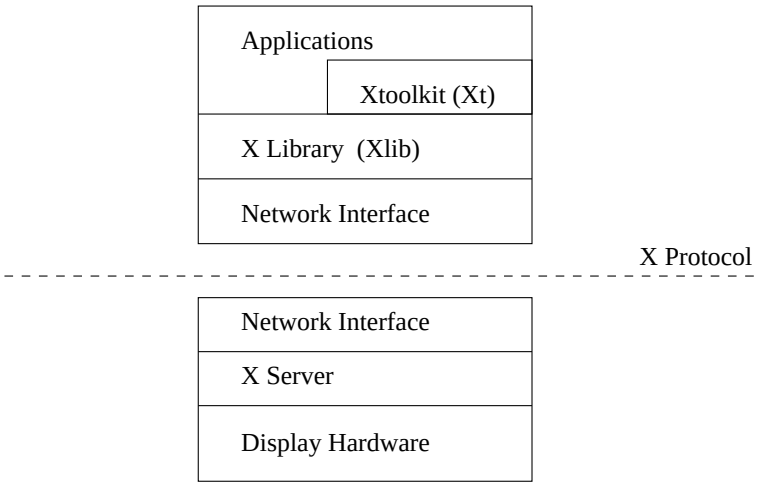

Figure 9.2: Architecture of the X System

programming environment. The application programmer can easily develop the user interface by dragging the required component types (e.g. menu, forms, etc.) from the displayed icons and placing them wherever required. Thus, visual programming can be considered as program development through manipulation of several visual objects. Reuse of program components in the form of visual objects is an important aspect of this style of programming.

19. What do you understand by component-based user interface development? What are the advantages of component-based user interface development?
Ans:
In this style of user interface development, every user interface can easily be built from a handfuls of predefined components such as menus, dialog boxes, forms, etc. Besides the standard components, and the facilities to create good interfaces from them, one of the basic support available to the user interface developers is the window system. The window system lets the application programmer create and manipulate windows without having to write the basic windowing functions.

20. Design and develop a graphical user interface for the graphical editor software described in problem 6.18.

21. Prepare a check list for user interface inspection.
Ans:

Visibility of the system status. The system should as far as possible keep the user informed about the status of the system and what is going on.

Match between the system and the real world. The system should speak the user's language with words, phrases, and concepts familiar to that used by the user, rather than using system-oriented terms.

Undoing mistakes. The user should feel that he is in control rather than feeling helpless or to be at the control of the system. An important step toward this is that the users should be able to undo and redo operations.

Consistency. The users should not have to wonder whether different words, concepts, and operations mean the same thing in different situations.

Recognition rather than recall. The user should not have to recall information which was presented in another screen. All data and instructions should be visible on the screen for selection by the user.

Support for multiple skill levels. Provision of accelerators for experienced users allows them to efficiently carry out the actions they most frequently require to perform.

Aesthetic and minimalist design. Dialogs should not contain information which are irrelevant and are rarely needed. Every extra unit of information in a dialog competes with the relevant units and diminishes their visibility.

Help and error messages. These should be expressed in plain language (no codes), precisely indicating the problem, and constructively suggesting a solution.

Error prevention. Error possibilities should be minimized. A key principle in this regard is to prevent the user from entering wrong values. In situations where a choice has to be made from among a discrete set of values, the control should present only the valid values using a drop-down list, a set of option buttons or a similar multichoice control. When a specific format is required for attribute data, the entered data should be validated when the user attempts to submit the data.

22. Study the user interface of some popular software products such as Word, Powerpoint, Excel, OpenOffice, Gimp, etc. and identify the metaphors used. Also, examine how tasks are broken up into subtasks and closure achieved.

23. What do you understand by a metaphor in a user interface design? Why is a metaphor-based user interface design advantageous? List a few metaphors which can be used for user interface design.
Ans:
The abstractions of real-life objects or concepts from day-to-day real-life examples used in user interface design are called metaphors. Metaphors help in interface development at lower effort and reduced costs for training the users. Over time, people have developed efficient methods of dealing with some commonly occurring situations. These solutions are the themes of the metaphors. Metaphors can also be based on physical objects such as a visitor's book, a catalog, a pen, a brush, a scissor, etc. A solution based on metaphors is easily understood by the users, reducing learning time and training costs. Some commonly used metaphors are the following:

- White board
- Shopping cart
- Desktop
- Editor's work bench
- White page
- Yellow page
- Office cabinet
- Post box
- Bulletin board
- Visitor's Book

24. What do you understand by a *modal dialog*? Why are these required? Why should the use of too many modal dialogs in an interface design be avoided?
Ans:
Some of the windows have to be defined as modal dialogs. When a window is a modal dialog, no other windows in the application is accessible until the current window is closed. When a modal dialog is closed, the user is returned to the window from which the modal dialog was invoked. Modal dialogs are commonly used when an explicit confirmation or authorization step is required for an action (e.g. confirmation or delete). Though use of modal dialogs are essential in some situations, overuse of modal dialogs reduces user flexibility. In particular, sequences of modal dialogs should be avoided.

25. How does the human cognition capabilities and limitations influence human-computer user interface designing?
Ans:
The following are some important issues concerning limitations of human cognition capabilities and its influence on human-computer user interface design.

Limited memory. Humans can remember at most seven unrelated items of information for short periods of time. Therefore, the GUI designer should not require the user to remember too many items of information at a time.

Frequent task closure. Doing a task (except for very trivial tasks) requires doing several subtasks. When the system gives a clear feedback to the user that a task has been successfully completed, the user gets a sense of achievement and relief.

Recognition rather than recall. Information recall incurs a larger memory burden on the users and is to be avoided as far as possible. On the other hand, recognition of information from the alternatives shown to him is more acceptable.

Procedural versus object-oriented. Procedural designs focus on tasks, prompting the user in each step of the task, giving them few options for anything else. This approach is best applied in situations where the tasks are narrow and well-defined or where the users are inexperienced, such as an ATM. An object-oriented interface on the other hand focuses on objects. This allows the users a wide range of options.

26. Distinguish between a user-centric interface design and interface design by users.
Ans:
The following are the important differences between user-centric interface design and interface design by users.

- User-centered design is the theme of almost all modern user interface design techniques. However, user-centered design does not mean design by users. One should not get the users to design the interface, nor should one assume that the user's opinion of which design alternative is superior is always right.
- Users have good knowledge of the tasks they have to perform, they also know whether they find an interface easy to learn and use but they have less understanding and experience in GUI design than the GUI developers.

Chapter 10

Coding and Testing

1. Distinguish between error and failure. Testing detects which of these two? Justify your answer.
   Ans:
   A **failure** is a manifestation of an **error** (or *defect* or *bug).*

   Testing detects failures. The exact cause for failure (i.e. error) has to be determined through debugging.

2. What are driver and stub modules in the context of integration and unit testing of a software product? Why are stub and driver modules required?
   Ans:
   Modules required to provide the necessary environment (which either call or are called by the module under test) are usually not available until they too have been unit tested, *stubs* and *drivers* are designed to provide the complete environment for a module. The role of stub and driver modules is pictorially shown in Fig. 10.1. A stub procedure is a dummy procedure that has the same I/O parameters as the given procedure but has a highly simplified behavior. For example, a stub procedure may produce the expected behavior using a simple table lookup mechanism. A driver module would contain the nonlocal data structures accessed by the module under test, and would also have the code to call the different functions of the module with appropriate parameter values.

3. State **TRUE** or **FALSE** of the following. Support your answer with proper reasoning:

   (a) The effectiveness of a test suite in detecting errors in a system

can be determined by counting the number of test cases in the suite.
Ans:
FALSE.
If the test cases detect the same errors, then a simple count of test cases is misleading.

(b) Once the McCabe's Cyclomatic complexity of a program has been determined, it is very easy to identify all the linearly independent paths of the program.
Ans:
FALSE.

(c) Use of static and dynamic program analysis tools is an effective substitute for thorough testing.
Ans:
FALSE.

(d) During code review you detect errors whereas during code testing you detect failures.
Ans:
TRUE.
During code review errors are directly spotted on the code whereas during code testing failures are detected and debugging is necessary to identify errors.

(e) A pure top-down integration testing does not require the use of any stub modules.
Ans:
FALSE.
A pure top-down integration testing does not require the use of any driver modules.

(f) Adherence to coding standards is checked during the system testing stage.
Ans:
FALSE.
Adherence to coding standards is checked before unit testing through code inpections.

(g) A program can have more than one linearly independent path.
Ans:
TRUE.
The upperbound on the number of linearly independent paths of a program is the Cyclomatic complexity metric for the program.

(h) The number of test cases required for branch coverage-based testing of a program can be greater than those required for path coverage-based testing of the same program.
Ans:
FALSE.
The number of test cases required for path coverage-based testing of a program can be greater than those required for branch coverage-based testing of the same program.

(i) Branch coverage-based testing is a stronger testing strategy compared to path coverage-based testing.
Ans:
FALSE.
Path coverage-based testing can detect errors that are not detected by branch coverage-based testing.

(j) Out of all types of internal documentation (i.e. provided in the source code), careful commenting is the most useful.
Ans:
FALSE.
Out of all types of internal documentation (i.e. provided in the source code), meaningful variable names is the most useful.

(k) Error and failure are synonymous in software testing terminology.
Ans:
FALSE.
A **failure** is a manifestation of an **error** (or *defect* or *bug).*

(l) Development of suitable driver and stub functions are essential for carrying out effective system testing of a product.
Ans:
FALSE.
System testing is carried out on a fully integrated system, therefore no driver or stub modules are necessary during system testing.

(m) System testing can be considered as a white box testing.
Ans:
FALSE.
System test cases test the functionality of the system as specified in the SRS document, and therefore are pure black-box tests.

(n) The main purpose of integration testing is to find design errors.
Ans:
FALSE.

The main purpose of integration testing is to find interface errors among modules.

(o) Introduction of additional edges and nodes in the CFG due to introduction of sequence type of statements in the program can increase the cyclomatic complexity of the program.
Ans:
FALSE.
Introduction of additional edges and nodes in the CFG due to introduction of sequence type of statements in the program has no effect on the cyclomatic complexity of the program.

(p) The terms software verification and software validation are essentially synonyms.
Ans:
FALSE.
Software verification concerns checking whether the output of one phase conforms to its preceding phase whereas software validation checks whther the fully functional system conforms to the specification.

(q) Code walkthrough for a module is normally carried out after unit test is over.
Ans:
FALSE.
Code walkthrough is carried out just before unit test.

(r) Code walkthrough for a module is normally carried out after the module succesfully compiles.
Ans:
TRUE.

(s) During code walkthrough most of the syntax errors are identified.
Ans:
FALSE.
Code walkthrough is carried out only after the code successfully compiles.

4. What is the difference between black-box testing and white-box testing?
Ans:
In black-box testing, test cases are designed from an examination of the input/output values only and no knowledge of design, or code is required. On the other hand, designing white-box test cases requires thorough knowledge about the internal structure of software.

5. What is the difference between internal and external documentation. What are the different ways of providing internal documentation? Out of these, which is most useful?
   Ans:
   Internal documentation is the code comprehension features provided as part of the source code itself. All other types of documents are called external documentation. Internal documentation is also provided through the use of meaningful variable names, module and function headers, code indentation, code structuring, use of enumerated types and constant identifiers, use of user-defined data types, etc.

6. What is meant by structural complexity of a program? Define a metric for measuring the structural complexity of a program. How is structural complexity of a program different from its *computational complexity*?
   Ans:
   The structural complexity of a program defines an upper bound on the number of independent paths in a program.

   Given a control flow graph G of a program, the structural complexity V(G) can be computed as:

   $$V(G) = E - N + 2$$

   where, N is the number of nodes of the control flow graph and E is the number of edges in the control flow graph.

7. Write a C function for searching an integer value from a large sorted sequence of integer values stored in an array of size 100, using the the binary search method.

   (a) Build the control flow graph of your binary search function, and hence determine its cyclomatic complexity.

   (b) Design a test suite for testing your binary search function.

   (c) How is cyclomatic metric useful in designing test suite for path coverage?

8. Given a software product and its requirements specification document, explain how would you design the system test suites for this software product.
   Ans:
   The functional test suite can be designed by examining the input

and output data domains of the functional requirements. The performance test suite can be designed by examining the non-functional requirements.

9. What is a coding standard? Identify the problems that might occur if the engineers of an organization do not adhere to any coding standard?
   Ans:
   A coding standard lists several rules to be followed during coding, such as the way variables are to be named, the way the code is to be laid out, error return conventions, etc.

   If no coding standard is used, the codes may be difficult to understand and may not be as efficient as code written using some well-defined coding standard.

10. What is the difference between a coding standard and a coding guideline? Why are these considered important in a software development organization? Write down five important coding standards and coding guidelines that you would recommend.
    Ans:
    Coding standards are followed compulsorily. Whereas coding guidelines provide general suggestions regarding the coding style to be followed but leave the actual implementation of these guidelines to the discretion of the individual engineers.
    Coding Standards.

    - *Rules for limiting the use of globals.*
    - *Contents of the headers preceding codes for different modules.*
    - *Naming conventions for global variables, local variables, and constant identifiers.*
    - *Error return conventions and exception handling mechanisms.*

    Coding Guidelines.

    - *Do not use a coding style that is too clever or too difficult to understand.*
    - *Avoid obscure side effects.*
    - *Do not use an identifier for multiple purposes.*
    - *The code should be well-documented.*

- *The length of any function should not exceed 10 source lines.*
- *Do not use goto statements.*

11. When during the development process is the compliance with coding standards is checked? List two coding standards each for (i) enhancing readability of the code, (ii) reuse of the code.
Ans:
Compliance with coding standrs is checked after the coding for a module is complete and the module successfully compiles. **Coding Standards for enhancing readability.**

    - *Contents of the headers preceding codes for different mod ules.*
    - *Naming conventions for global variables, local variables, and constant identifiers.*

    Coding Standards for enhancing reuse.

    - *Rules for limiting the use of globals.*
    - *Error return conventions and exception handling mechanism s.*

12. Distinguish between software verification and software validation. When during the software life cycle are verification and validation performed? Can one be used in place of the other?
Ans:
Software verification concerns checking whether the output of one phase conforms to its preceding phase whereas software validation checks whther the fully functional system conforms to the specification.

    Verification is performed at the end of every phase whereas validation is performed after a fully integrated and working product is ready.

13. Which one of the following is the strongest structural test technique: statement coverage-based testing, branch coverage-based testing, or condition coverage-based testing? Justify your answer with the help of a suitable example.
Ans:
Condition coverage is the strongest structural test technique.

14. Prove that branch coverage testing is stronger than statement coverage testing.
Ans:
It can be proved by proving:

(a) Branch coverage ensures statement coverage

(b) Branch coverage can detect some errors that cannot be detected using statement coverage.

The first part can be proved by noting that each statement of a program belongs to some branch. So, when branch coverage is achieved statement coverage is automatically achieved.

The second part can be proved by the fact that staement coverage cannot detect error in a program containing a loop which manifests only when the loop condition is false.

15. Which is a stronger testing: Data flow testing or path testing? Give the reasoning behind your answer.

16. Briefly highlight the difference between code inspection and code walk through. Compare the relative merits of code inspection and code walk through.
Ans:
In contrast to code walk through, the aim of code inspection is to discover some common types of errors caused due to oversight and improper programming. In other words, during code inspection the code is examined for the presence of certain kinds of errors, in contrast to the hand simulation of code execution done in code walk throughs.

17. What is meant by a code walk through? What are some of the important types of errors checked during code walk throughs?
Ans:
In code walk through, each member of the review team selects some test cases and simulates execution of the code by hand (i.e. trace execution through each statement and function execution). The main objectives of the walk through are to discover the algorithmic and logical errors in the code.

18. Answer the following:

(a) Suppose a program contains N decision points, each of which has two branches. How many test cases are necessary for branch testing?

(b) If there are M choices at each decision point, how many test cases are needed for branch testing? Is it possible to achieve branch coverage using a smaller number of test cases than you

have answered depending on the branch conditions? Justify your answer.

(c) For a program containing N binary branches how many test cases are necessary for path coverage? For a program containing N number of M-ary branches, how many test cases are necessary for path coverage?

(d) How many test cases are necessary for path testing in each case? Justify your answer.

Ans:

(a) 2*n test cases.

(b) m*n test cases. Yes, if some branches cannot assume certain logic values, then it is possible to achieve branch coverage using smaller number of test cases.

(c) n+1 test cases are needed for path coverage as the cyclomatic complexity is n+1 For m-are branches, n*m test cases are needed.

19. Usually large software products are tested at three different testing levels, i.e., unit testing, integration testing, and system testing. Why not do a single thorough testing of the system after the system is developed, e.g. detect all the defects of the product during system testing?
Ans:
If there is only one level of testing then there would be too many places to look for an error and debuggin effort would be very high.

20. What do you understand by the term system testing? What are the different kinds of system testing that are usually performed on large software products?
Ans:
System tests are designed to validate a fully developed system to assure that it meets its requirements. There are essentially three main kinds of system testing:

(a) **Alpha Testing.** Alpha testing refers to the system testing carried out by the test team within the developing organization.

(b) **Beta Testing.** Beta testing is the system testing performed by a select group of friendly customers.

(c) **Acceptance Testing.** Acceptance testing is the system testing performed by the customer to determine whether he should accept the delivery of the system.

21. Usability of a software product is tested during which type of testing: unit, integration, or system testing? How is usability tested?
Ans:
Usability of a product is tested during system testing. During usability testing, the isplay screens, messages, report formats, and other aspects relating to the user interface requirements are tested.

22. Suppose a developed software has successfully passed all the three levels of testing, i.e. unit testing, integration testing, and system testing. Can we claim that the software is defect free? Justify your answer.
Ans:
No, we still cannot claim that the software is defect free. The three levels of testing do not do an exhaustive testing of the program using all possible data values.

23. Distinguish between the static and dynamic analysis of a program. How are static and dynamic program analysis results useful?
Ans:
Static analysis tools assess and compute various characteristics of a software product without executing it. Dynamic program analysis techniques require the program to be executed and its actual behavior recorded. Static analysis tools can indicate whether adequate commenting has been done, whether some coding standards have been adhered to or not.

Dynamic analysis tools are normally used to check whether testing is satisfactory.

24. What do you understand by automatic program analysis? Give a broad classification of the different types of program analysis tools used during program developement. What are the different types of information produced by each type of tool?
Ans:
Automated program analysis takes the source code or the executable code of a program as input and produces reports regarding several important characteristics of the program, such as its size, complexity, adequacy of commenting, adherence to programming standards,

etc. Also, some program analysis tools produce reports regarding the adequacy of the test cases. We can classify all these tools into two broad categories of program analysis tools:

- Static analysis tools
- Dynamic analysis tools

25. Design black-box test suites for a function that checks whether a character string (of up to 25 characters length) is a palindrome.
Ans:
The equivalence classes are:

- Palindrome
- Not a palindrome
- Null string
- String containing more than 25 charcters.

The equivalence class tests can be designed by choosing one representative test case from each equivalence class. The boundary value can be checked by selecting a word with 25 characters and another word with 26 characters.

26. Design the black-box test suite for a function that takes the name of a book as input and searches a file containing the names of the files available in the Library and displays the details of the book if the book is available in the library otherwise displays the message "book not available".
Ans:
The equivalence classes are:

- book available
- book not available
- invalid book name

The equivalence class tests can be designed by choosing one representative test case from each equivalence class.

27. Why is it important to properly document a software product? What are the different ways of documenting a software product?
Ans:
Proper documents facilitate maintenance. The 2 important types of documents are internal documentation and external documentation.

28. Why is it advantageous to detect errors during code and design reviews rather than detecting these errors at the time of testing?
Ans:

It costs less, since testing detects failures and additional debugging effort is needed to identify the error causing the failure.

29. What do you understand by the clean room strategy? What are its advantages?
Ans:
The clean room strategy relies heavily on walk throughs, inspection, and formal verification. The programmers are not allowed to test any of their code by executing the code other than doing some syntax testing using a compiler.

This technique reportedly produces documentation and code that is more reliable and maintainable than other development methods relying heavily on code execution-based testing.

30. How can you compute the cyclomatic complexity of a program? How is cyclomatic complexity useful in program testing?
Ans:
Given a control flow graph G of a program, the cyclomatic complexity V(G) can be computed as:

$$V(G) = E - N + 2$$

where, N is the number of nodes of the control flow graph and E is the number of edges in the control flow graph.

Cyclomatic complexity metric identifies an upperbound on the number of independednt paths. This helps to determine whether path testing is adequately done or not.

31. Suppose in order to estimate the number of latent errors in a program, you seed it with 100 errors of different kinds. After testing the software using its full test set, you discover only 80 of the introduced errors. You discover 15 other errors also. Estimate the number of latent errors in the software. What are the limitations of the error seeding method?
Ans:
Number of latent errors=(100*15/80 - 15)=19-15=4

The limitation of the error seeding method is that it implicitly assumes that the seeded errors cloasesly math with the type and frequency of the existing errors.

32. What is stress testing? Why is stress testing applicable to only certain types of systems?
Ans:
Stress testing evaluates system performance when it is stressed for short periods of time.

Stress testing is applicable to only those products which have an element of time or size in the specification of the system.

33. What do you understand by unit testing? Write the code for a module that contains the functions to implement the functionality to implement a bounded stack of 100 integers. Assume that the stack functions supported are push, pop, and is-empty.

34. What do you understand by the term integration testing? Which types of defects are uncovered during integration testing? What are the different types of integration testing methods that can be used to carry out integration testing of a large software product? Compare the merits and demerits of these different integration testing strategies.
Ans:

Interface defects among modules are detected during integration testing.

35. Discuss how you would perform system testing of a software that implements a queue of positive integral elements. Assume that the queue supports only the functions insert an element, delete an element, and find an element.

36. Why is it advantageous to detect as many error as possible during code review than during testing?
Ans:
It costs less, since testing detects failures and additional debugging effort is needed to identify the error causing the failure.

37. What do you mean by side effects of a function call? Give examples of side effects. Why are obscure side effects undesirable?

Ans:
The side effects of a function call include modification of parameters passed by reference, modification of global variables, and I/O operations. An obscure side effect is one that is not obvious from a casual examination of the code. Obscure side effects make it difficult to understand a piece of code.

38. What is regression testing? When is regression testing done? Why is regression testing necessary? How is regression testing performed?
Ans:
Regression testing is the practice of running an old test suite after each change to the system or after each bug fix to ensure that no new bug has been introduced due to the change or the bug fix. However, if only a few statements are changed, then te entire test suite need not be run — only those test cases that test the functions that are likely to be affected by the change need to be run.

39. Do you agree with the following statement: "System testing can be considered as a pure black-box test." Justify your answer.
Ans:
Yes. System test cases are designed based on the functional and non-functional requirements specified in the SRS document.

40. What do you understand by big-bang integration testing? How is big-bang integration testing performed? Describe situations where big-bang integration testing is desirable.
Ans:
In big-bang integration all the modules are integrated in a single step. Big-bang testing is desirable in very small systems having only 3 or 4 modules.

41. What is the relationship between cyclomatic complexity and program comprehensibility? Can you justify why such an apparent relationship exists?
Ans:
Program comprehensibility decreases as the cyclomatic complexity increases.

    The reason for this is the fact that cyclomatoc complexity is an upperbound on the number of independent paths in the program. Programmers generally understand the working of a program by tracing the paths from the outputs to inputs. When the cyclomatic complexity becomes high, the number of paths to be traced increases

and consequently it becomes much more difficult to understand the program.

42. Consider the following C function named `bin-search`:

```
/* num is the number the function searches in a presorted integer array arr */
   int bin_search(int num)
      {
         int min,max;

         min =0;
         max =100;
         while(min!=max){
            if(arr[(min+max)/2]>num)
                max=(min+max)/2;
            else if(arr[(min+max)/2]<num)
            min=(min+max)/2;
            else return((min+max)/2);
        }
        return(-1);
   }
```

Design a test suite for the function `bin-search` using the following white-box testing strategies (Show the intermediate steps in deriving the test cases):

- statement coverage;
- branch coverage
- condition coverage
- path coverage

Ans:

43. Consider the following C function named sort.

```
/* sort takes an integer array and sorts it in ascending order */
void sort(int a[], int n){
   int i,j;

   for(i=0;i<n-1;i++)
      for(j=i+1;j<n;j++)
```

```
            if(a[i]>a[j])
            {
               temp=a[i];
               a[i]=a[j];
               a[j]=temp;
            }
}
```

(a) Determine the cyclomatic complexity of the sort function.

(b) Design a test suite for the function `sort` using the following white-box testing strategies (Show the important steps in your test suite design method).

- statement coverage
- branch coverage
- condition coverage
- path coverage

44. Draw the control flow graph for the following i function named `find-maximum`. From the control flow graph, determine its Cyclomatic complexity.

```
int find-maximum(int i,int j, int k)
{
     int max;

     if(i>j) then
         if(i>k) then max=i;
                    else max=k;
         else  if(j>k) max=j
         else max=k;
       return(max);
  }
```

Ans:
The control flow graph is shown in Fig. 1.

The cyclomatic complexity is = 11-8+1=4
It can be also computed as number of independent regions+1
=3+1=4

```
int  find−maximum(int  i,int j,int k){

int max;

1     if(i>j) then
2        if(i>k)  then
3                max=i;
4               else  max=k;
5        else if(j>k)
6             max=j;
7             else  max=k;
8     return(max);
}
```

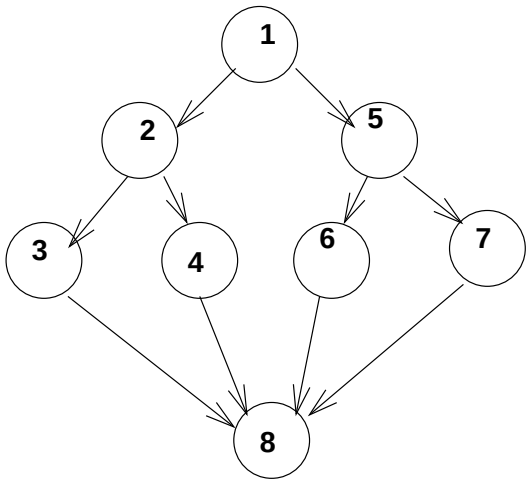

Figure 10.1: Control Flow Graph for the Problem

45. When during the life cycle are the compliance to the coding standards and guidelines are checked?
Ans:
Compliance to coding standards and guidelines is checked before unit testing and after the code successfully compiles.

46. Identify the types of defects that you would be able to detect during the following:

    (a) code inspection

    (b) code walk through

Ans:

(a) code inspection: during code inspection compliance with coding standards and presence of standard types of errors is checked.

(b) code walk through: during code walk through, presence of logical errors is checked.

47. Design the black-box test suite for a function named **quadratic-solver**. Quadratic-solver accepts three floating point numbers (a,b,c) representing a quadratic equation of the form $ax^2 + bx + c = 0$. It computes and displays the solution.
Ans:
The equivalence classes for the quadratic solver are the following:

    - Real solutions
    - Imaginary solutions
    - No solutions: a=b=0
    - Coincident solutions

    The equivalence class tests can be designed by choosing one representative test case from each equivalence class.

48. Design the black-box test suite for a function that accepts two pairs of floating point numbers representing two coordinate points. Each pair of coordinate points represents the center and a point on the circumference of the circle. The function prints whether the the two circles are intersecting, one is contained within the other, or are disjoint.
Ans:
The equivalence classes for the given function are the following:

    - circles intersecting at 2 points
    - circles intersecting at 1 point
    - disjoint circles
    - circles intersect at infinite number of points (coincident circles)
    - invalid circle parameters

    The equivalence class tests can be designed by choosing one representative test case from each equivalence class.

49. Design black-box test suites for a function called **find-intersection**. The function **find-intersection** takes four real numbers m_1, c_1, m_2, c_2 as its arguments representing two straight lines $y = m_1x + c_1$ and $y = m_2x + c_2$. It determines the points of intersection of the two lines. Depending on the input values to the function, it displays any one of the following messages:

    - single point of intersection
    - overlapping lines — infinite points of intersection

- parallel lines — no points of intersection
- invalid input values

Ans:
The equivalence classes for the given function are the following:

- single point of intersection
- overlapping lines — infinite points of intersection
- parallel lines — no points of intersection
- invalid input values

The equivalence class tests can be designed by choosing one representative test case from each equivalence class.

50. Design black-box test suite for the following progam. The program accepts two pairs of coordinates (x1,y1),(x2,y2), (x3,y3), (x4,y4). The first two points (x1,y1) and (x2,y2) represent the lower left and the upper right points of the first rectangle. The second two points (x3,y3) and (x4,y4) represent the lower left and the upper right points of the second rectangle. It is assumed that the length and width of the rectangle are parallel to either the x-axis or y-axis. The program computes the points of intersection of the two rectangles and prints their points of intersection.
Ans:
The equivalence classes for the given function are the following:

- disjoint rectangles
- intersecting rectangles
- rectangles with one side overlapping
- rectangles touching at one point
- concident rectangles
- rectangles with two sides overlapping

The equivalence class tests can be designed by choosing one representative test case from each equivalence class.

51. Design black-box test suite for a program that accepts upto 10 simultaneous linear equations in upto 10 independent variables and displays the solution.
Ans:
The equivalence classes for the given function are the following:

- unique solutions
- infinite solutions
- no solutions
- too many variables
- invalid equations (e.g. 5=0)

The equivalence class tests can be designed by choosing one representative test case from each equivalence class.

52. Design the black-box test suite for the following Library Automation Software. The Library Automation Software accepts a string representing the name of a book. It checks the library catalog, and displays whether the book is listed in the catalog or not. If the book is listed in the catalog, it displays the number of copies that are currently available in the racks and the copies issued out.
Ans:
The equivalence classes for the given function are the following:

    - book available
    - book not available
    - book issued out

    The equivalence class tests can be designed by choosing one representative test case from each equivalence class.

53. Design the black-box test suite for a program that accepts two strings and checks if the first string is a substring of the second string and displays the number of times the first string occurs in the second string.
Ans:
The equivalence classes for the given function are the following:

    - no match
    - identical strings
    - proper substring

    The equivalence class tests can be designed by choosing one representative test case from each equivalence class.

54. Design black-box test suite for a program that accepts a pair of ponts defining a straight line and another point and a float number defining the center of a circle and its radius. It computes their points of

intersection and prints them.
Ans:
The equivalence classes for the given function are the following:

- intersecting at one point
- intersecting at two points
- no points of intersection
- invalid input

The equivalence class tests can be designed by choosing one representative test case from each equivalence class.

55. Among the different development phases of life cycle, testing typically requires the maximum effort. Identify the main reasons behind the large effort necessary for this phase.

56. What do you understand by performance testing? What are the different types of performance testing that should be performed for each of the problems 10–20 of Chapter 5?

57. Identify the types of information that should be presented in the test summary report.
Ans:
Test summary report normally covers each subsystem and represents a summary of tests which have been applied to the subsystem. It will specify how many tests have been applied to a subsystem, how many tests have been successful, how many have been unsuccessful, and the degree to which they have been unsuccessful, e.g. whether a test was an outright failure or whether some of the expected results of the test were actually observed.

58. What is the difference between the top-down and the bottom-up integration testing approaches? Explain your answer using an example. Why is the mixed integration testing approach preferred by many testers?
Ans:

Chapter 11

Software Reliability and Quality Management

1. Identify the factors which make the measurement of software reliability a much harder problem than the measurement of hardware reliability.
   Ans:
   Most of the hardware failures are caused by wear and tear, whereas all software failures are due to design and coding errors. Therefore, the reliability of hardware is stable after repair whereas reliability of software changes after each error fix.

2. Discuss how the reliability changes over the life time of a software product and a hardware product.
   Ans:
   For a hardware product, the reliability appears like the shape of a bath tub over its life time. On the other hand, software reliability over a large period of time continually improves. This behavior of reliability is shown in figure 1.

3. Explain using one simple sentence each what you understand by the following reliability measures:

   - A POFOD of 0.001
     Ans:
     POFOD of 0.001 means that 1 out of every 1000 service requests is expected to result in a failure.
   - A ROCOF of 0.002
     Ans:

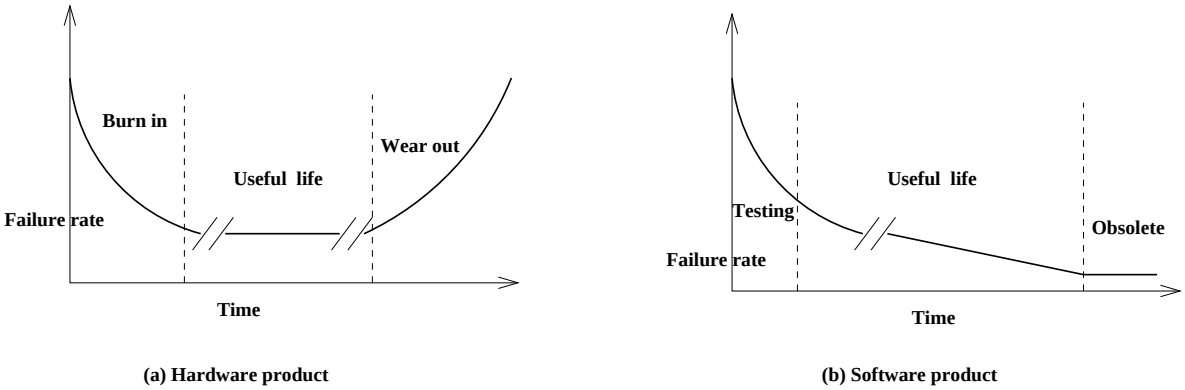

Figure 11.1: Change in failure rate of a product

A ROCOF of 0.002 means that 2 failures are expected over 1000 units of time.

- MTBF of 200 units
  Ans:
  MTBF of 200 units indicates that once a failure occurs, the next failure is expected after 200 time units. In this case, the time measurements are real time and not the execution time as in MTTF.
- Availability of 0.998
  Ans:
  Availability of 0.998 indicates that over a unit time interval, the chances of the system operating without failure is 99.8

4. Define three metrics to measure software reliability. Do you consider these metrics entirely satisfactory to provide measure of the reliability of a system? Justify your answer.

5. How can you determine the number of latent defects in a software product during the testing phase?
   Ans:
   The number of latent defects can be determined using the error seeding technique.

6. State **TRUE** or **FALSE** of the following. Support your answer with proper reasoning:

(a) The reliability of a software product increases almost linearly, each time a defect gets detected and fixed.
Ans:
FALSE
Each time an error is fixed, the reliability may not grow by a constant amount due to the following reasons:

- Imperfect error fix: An error fix may result in additional errors.
- Diminishing returns: As errors are detected and fixed, the increase in reliability reduces.

(b) As testing continues, the rate of growth of reliability slows down representing a diminishing return of reliability growth with testing effort.
Ans:
TRUE
The errors whic frequently result in failures are fixed first. Therefore, the initial errors to be fixed result in higher reliability growth than the errors detected and fixed later.

(c) Modern quality assurance paradigms are centered around carrying out thorough product testing.
Ans:
FALSE
Modern quality assurance paradigms are centered around the concept that if a good process is rigorously followed then the end product is bound to be good.

(d) An important use of receiving a ISO 9001 certification by a software organization is that it can improve its sales efforts by advertizing its products as conforming to ISO 9001 certification.
Ans:
FALSE
The ISO certification is awarded with the explicit understanding that it can be used only corporate advertisements and not product advertizements as the certificate is not given to any specific product but to the process.

7. What does the quality parameter "fitness of purpose" mean in the context of software products? Why is this not a satisfactory criterion for determining the quality of software products?
Ans:
The modern view of a quality associates with a software product

several quality factors such as portability, usability, reusability, correctness, maintainability, etc.

To give an example why fitness of purpose is not a satisfactory criterion for determining the quality of software products, consider a software product that is functionally correct. That is, it performs all functions as specified in the SRS document. But, has an almost unusable user interface.

8. Why is it important for a software development organization to obtain ISO 9001 certification?
Ans:
The benefits that accrue to organizations obtaining ISO certification.

   - Confidence of customers in an organization increases when the organization qualifies for ISO 9001 certification. This is especially true in the international market. In fact, many organizations awarding international software development contracts insist that the development organization have ISO 9000 certification. For this reason, it is vital for software organizations involved in software export to obtain ISO 9000 certification.
   - ISO 9000 requires a well-documented software production process to be in place. A well-documented software production process contributes to repeatable and higher quality of the developed software.
   - ISO 9000 makes the development process focused, efficient, and cost-effective.
   - ISO 9000 certification points out the weak points of an organizations and recommends remedial action.
   - ISO 9000 sets the basic framework for the development of an optimal process and TQM.

9. Discuss the relative merits of ISO 9001 certification and the SEI CMM-based quality assessment.
Ans:

   - ISO 9000 is awarded by an international standards body. Therefore, ISO 9000 certification can be quoted by an organization in official documents, communication with external parties, and in tender quotations. However, SEI CMM assessment is purely for internal use.

- SEI CMM was developed specifically for software industry and therefore addresses many issues which are specific to software industry alone.
- SEI CMM goes beyond quality assurance and prepares an organization to ultimately achieve TQM. In fact, ISO 9001 aims at level 3 of SEI CMM model.
- SEI CMM model provides a list of key process areas (KPAs) on which an organization at any maturity level needs to concentrate to take it from one maturity level to the next. Thus, it provides a way for achieving gradual quality improvement.

10. List five salient requirements that a software development organization must comply with before it can be awarded the ISO 9001 certificate. What are some of the shortcomings of the ISO certification process?
Ans:

- All documents concerned with the development of a software product should be properly managed, authorized, and controlled. This requires a configuration management system to be in place.
- Proper plans should be prepared and then progress against these plans should be monitored.
- Important documents should be independently checked and reviewed for effectiveness and correctness.
- The product should be tested against specification.
- Several organizational aspects should be addressed e.g., management reporting of the quality team.

11. With the help of suitable examples discuss the types of software organizations to which ISO 9001, 9002, and 9003 standards respectively are applicable.
Ans:

ISO 9001. This standard applies to the organizations engaged in design, development, production, and servicing of goods. This is the standard that is applicable to most software development organizations.

ISO 9002. This standard applies to those organizations which do not design products but are only involved in production. Examples of this category of industries include steel and car manufacturing industries who buy the product and plant designs from external sources and are involved in only manufacturing those products. Therefore, ISO 9002 is not applicable to software development organizations.

ISO 9003. This standard applies to organizations involved only in installation and testing of the products.

12. During software testing process, why is the reliability growth initially high but slows down later on?
Ans:
The errors whic frequently result in failures are fixed first. Therefore, the initial errors to be fixed result in higher reliability growth than the errors detected and fixed later.

13. If an organization does not document its quality system, what problems would it face?
Ans:
Without a properly documented quality system, the application of quality controls and procedures become *ad hoc*, resulting in large variations in the quality of the products delivered. Also, an undocumented quality system sends clear messages to the staff about the attitude of the organization towards quality assurance.

14. What according to you is a quality software product?
Ans:

15. Discuss the stages through which the quality system paradigm and the quality assurance methods have evolved over the years.
Ans:
The stages of evolution of the quality system paradigm are shown in Fig. 2.

16. Which standard is applicable to software industry, ISO 9001, ISO 9002, or ISO 9003?
Ans:
ISO 9001 is applicable.

17. In a software development organization, identify the persons responsible for carrying out the quality assurance activities. Explain the principal tasks they perform to meet this responsibility.

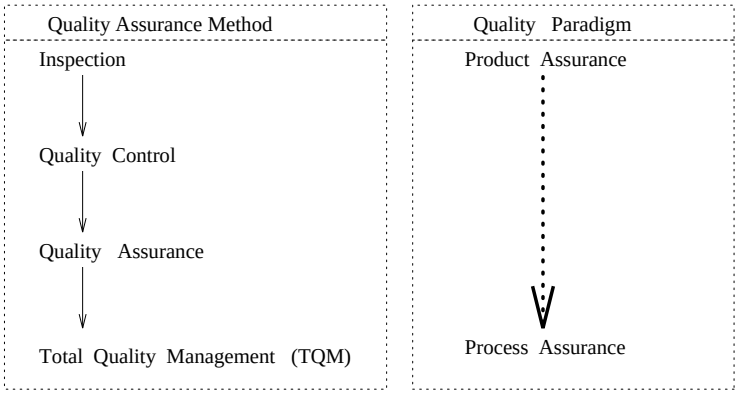

Figure 11.2: Evolution of quality system and corresponding shift in the quality paradigm

Ans:
Quality assurance is the responsibility of both the development team and the quality assurance team. The quality assurance team is generally responsible for:

- auditing of projects
- review of the quality system
- development of standards, procedures, and guidelines, etc.
- production of reports for the top management summarizing the effectiveness of the quality system in the organization.

The development team is responsible for reviews, testing, etc.

18. Suppose an organization mentions in its job advertizement that it has been assessed at level 3 of SEI CMM, what can you infer about the current quality practices at the organization? What does this organization have to do to reach SEI CMM level 4?
Ans:
The organization has a defined proccess, it has a well defined and meaningful training program, it carries out various reviews during the development process. To reach level 4 it has to concentrate on process and product metrics.

19. Suppose as the president of a company, you have the choice to either go for ISO 9000 based quality model or SEI CMM based model, which one would you prefer? Give the reasoning behind your choice.

Ans:
The choice would depend on various factors.

- SEI CMM provides a step by step path for quality improvement. If currently, the organization does not follow any quality policy then SEI CMM is preferable.
- If the company needs to bid for Governement contracts or work with European clients, the ISO 9001 certification is desirable.

20. What do you understand by total quality management (TQM)? What are the advantages of TQM? Does ISO 9000 standard aim for TQM?
Ans:
Total quality management (TQM) advocates that the process followed by an organization must be continuously improved through process measurements. TQM goes a step further than quality assurance and aims at *continuous process improvement.* TQM goes beyond documenting processes to optimizing them through redesign.

No, ISO 9001 does not aim for TQM.

21. What are the principal activities of a modern quality system?
Ans:
The quality system activities encompass the following:

- auditing of projects
- review of the quality system
- development of standards, procedures, and guidelines, etc.
- production of reports for the top management summarizing the effectiveness of the quality system in the organization.

22. In a software development organization whose responsibility is it to ensure that the products are of high quality? Explain the principal tasks they perform to meet this responsibility.
Ans:
It is the responsibility of the organization as a whole. Besides, reviews, testing etc. the other activities include the following:

- auditing of projects
- review of the quality system
- development of standards, procedures, and guidelines, etc.
- production of reports for the top management summarizing the effectiveness of the quality system in the organization.

23. What do you understand by repeatable software development? Organizations assessed at which level SEI CMM maturity achieve repeatable software development?
Ans:
Repeatable software development means that an organization can repeat its success in one project with similar projects. Organizations assessed at SEI CMM level 3 achieve repeatable software development.

24. What do you understand by Key Process Area (KPA), in the context of SEI CMM? Would there be any problem if an organization tries to implement higher level SEI CMM KPAs before achieving lower level KPAs? Justify your answer using suitable examples.
Ans:
Key Process Areas (KPAs) that indicate the areas an organization should focus to improve its software process to the next level.

SEI CMM provides a list of key areas on which to focus to take an organization from one level of maturity to the next. Thus, it provides a way for gradual quality improvement over several stages. Each stage has been carefully designed such that one stage enhances the capability already built up. For example, it considers that trying to implement a defined process (level 3) before a repeatable process (level 2) would be counterproductive as it becomes difficult to follow the defined process due to schedule and budget pressures.

25. What is the Six Sigma quality initiative? To which category of industries is it applicable? Explain the Six Sigma technique adopted by software organization with respect to the goal, the procedure, and the outcome.
Ans:
The purpose of Six Sigma is to improve processes to do things better, faster, and at lower cost. It can be used to improve every facet of business, from production, to human resources, to order entry, to technical support. Six Sigma can be used for any activity that is concerned with cost, timeliness, and quality of results.

The fundamental objective of the Six Sigma methodology is the implementation of a measurement-based strategy that focuses on process improvement and variation reduction through the application of Six Sigma improvement projects. This is accom plished through the use of two Six Sigma sub-methodologies: DMAIC and DMADV. The Six Sigma DMAIC process (define, measure, analyze, improve, control)

is an improvement system for existing processes falling below specification and looking for incremental improvement. The Six Sigma DMADV process (define, measure, analyze, design, verify) is an improvement system used to develop new processes or products at Six Sigma quality levels. It can also be employed if a current process requires more than just incremental improvement. Both Six Sigma processes are executed by Six Sigma Green Belts and Six Sigma Black Belts, and are overseen by Six Sigma Master Black Belts.

26. What is the difference between process metrics and product metrics? Give four examples of each.
Ans:
Product metrics measure the characteristics of the product being developed, such as its size, reliability, time complexity, understandability, etc. Process metrics reflect the effectiveness of theprocess being used, such as average defect correction time, productivity, average number of defects found per hour of inspection, average number of failures detected during testing per LOC, etc.

27. Suppose you want to buy a certain software product and you have kept a purchase precondition that the vendor must install the software, train your manpower on that, and maintain the product for at least a year, only then would you release the payment. Also, you do not foresee any maintenance requirement for the product once it works satisfactorily. Now, you receive bids from three vendors. Two of the vendors quote Rs. 3 Lakhs and Rs. 4 Lakhs whereas the third vendor quotes Rs 10 Lakhs saying that the prices would be high because they would be following a good development process as they have been assessed at the Level 5 of SEI CMM. Discuss how you would decide whom to award the contract.
Ans:

Since both the vendors undertake to maintain the product for a year and the business rules are not expected to change in future, the lower bid can be accepted.

Chapter 12

Computer Aided Software Engineering

1. What do you understand by the terms a CASE tool and a CASE environment? Why integration tools increases the power of the tools? Explain using some examples.
   Ans:
   A CASE tool is a generic term used to denote any form of automated support for software engineering, In a more restrictive sense a CASE tool can mean any tool used to automate some activity associated with software development.

   In a CASE environment a set of CASE tools are integrated into a common framework or environment. If the different CASE tools are not integrated, then the data generated by one tool would have to input to the other tools. This may also involve format conversions as the tools developed by different vendors are likely to use different formats. This results in additional effort of exporting data from one tool and importing to another. Also, many tools do not allow exporting data and maintain the data in proprietary formats.

2. What is a programming environment?
   Ans:
   A programming environment is an integrated collection of tools to support only the coding phase of software development. The tools commonly integrated in a programming environment are a text editor, a compiler, and a debugger. The different tools are integrated to the extent that once the compiler detects an error, the editor takes automatically goes to the statements in error and the error statements

are highlighted. Examples of popular programming environments are Turbo C environment, Visual Basic, Visual C++, etc.

3. What are the main advantages of using CASE tools?
   Ans:

   - A key benefit arising out of the use of a CASE environment is cost saving through all developmental phases. Different studies carry out to measure the impact of CASE put the effort reduction between 30% to 40%.
   - Use of CASE tools leads to considerable improvements to quality. This is mainly due to the facts that one can effortlessly iterate throug h the different phases of software development and the chances of human error is considerably reduced.
   - CASE tools help produce high quality and consistent documents. Since the important data relating to a software product are maintained in a central repository, redundancy in in the stored data is reduced and therefore chances of inconsistent documentation is reduced to a great extent.
   - CASE tools take out most of the drudgery in a software engineers work. For example, they need not check meticulously the balancing of the DFDs but can do it effortlessly through the press of a button.
   - CASE tools have led to revolutionary cost saving in software maintenance efforts. This arises not only due to the tremendous value of a CASE environment in traceability and consistency checks, but also due the systematic information capture during the various phases of software development as a result of adhering to a CASE environment.
   - Introduction of a CASE environment has an impact on the style of working of a company, and makes it oriented towards the structured and orderly approach.

4. What are some of the important features that a future generation CASE tool should support?
   Ans:

   Intelligent diagramming support. The fact that diagrammin g techniques are useful for system analysis and design is well established . The future CASE tools would provide help to aesthetically and automatically lay out

the diagrams.

Integration with implementation environment. The CASE tools should provide integration between design and implementation.

Data dictionary standards. The user should be allowed to integrate many development tools into one environment. It is highly unlikely that any one vendor will be able to deliver a total solution. Moreover, a preferred tool would require tuning up for a particular system. Thus the user would act as a system integrator. This is possible only if some standard on data dictionary emerges.

Customization support. The user should be allowed to define new types of objects and connections. This facility may be used to build some special methodologies. Ideally it should be possible to specify the rules of a methodology to a *rule engine* for carrying out the necessary consistency checks.

5. Discuss the role of the data dictionary in a CASE environment.
   Ans:
   Since different tools covering different stages share common information, it is required that they integrate through some central repository to have a consistent view of information associated with the software. This central repository is usually a data dictionary containing the definition of all composite and elementary data items.

6. Schematically draw the architecture of a CASE environment and explain how the different tools are integrated.
   Ans:
   The architecture of a CASE environment is shown in Fig. 1.

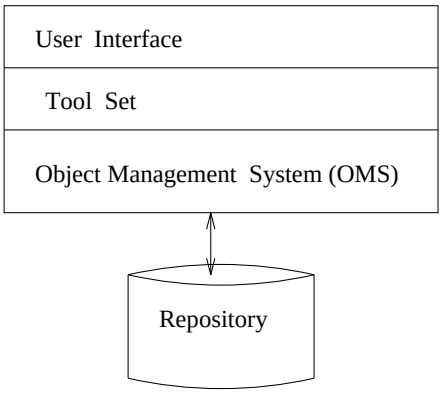

Figure 12.1: Architecture of a Modern CASE Environment

Chapter 13

Software Maintenance

1. What are the different types of maintenance that a software product might need? Why are these maintenance required?
   Ans:
   Software maintenance can be required for three main reasons:

   Corrective. Corrective maintenance of a software product is necessary either to rectify the bugs observed while the system is in use.

   Adaptive. A software product might need maintenance when the customers need the product to run on new platforms, on new operating systems, or when they need the product to interface with new hardware or software.

   Perfective. A software product needs maintenance to support the new features that users want it to support, to change different functionalities of the system according to customer demands, or to enhance the performance of the system.

2. Discuss the process models for software maintenance and indicate how you would select an appropriate maintenance model for a maintenance project at hand.
   Ans:
   The two process models for maintenance are shown in fugures 1 and2.

   An empirical study indicates that process 1 is preferable when the amount of rework is no more than 15% (see Fig. 13.5). Besides the amount of rework, several other factors might affect the decision regarding using process model 1 over process model 2:

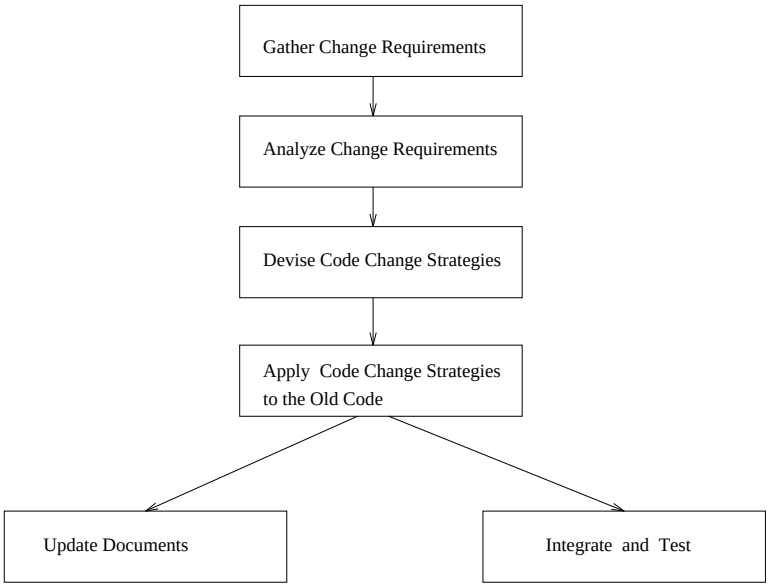

Figure 13.1: Maintenance process model 1

- Reengineering might be preferable for products which exhibit a high failure rate.
- Reengineering might also be preferable for legacy products having poor design and code structure.

3. State whether the following statements are **TRUE** or **FALSE**. Give reasons for your answer.

   (a) Legacy products are those products which have been developed long time back.
   Ans:
   FALSE
   Legacy software refers to those software products which are hard to maintain.

   (b) Corrective maintenance is the type of maintenance that is frequently carried out on a typical software product.
   Ans:
   FALSE
   Perfective maintenance is carried out most frequently.

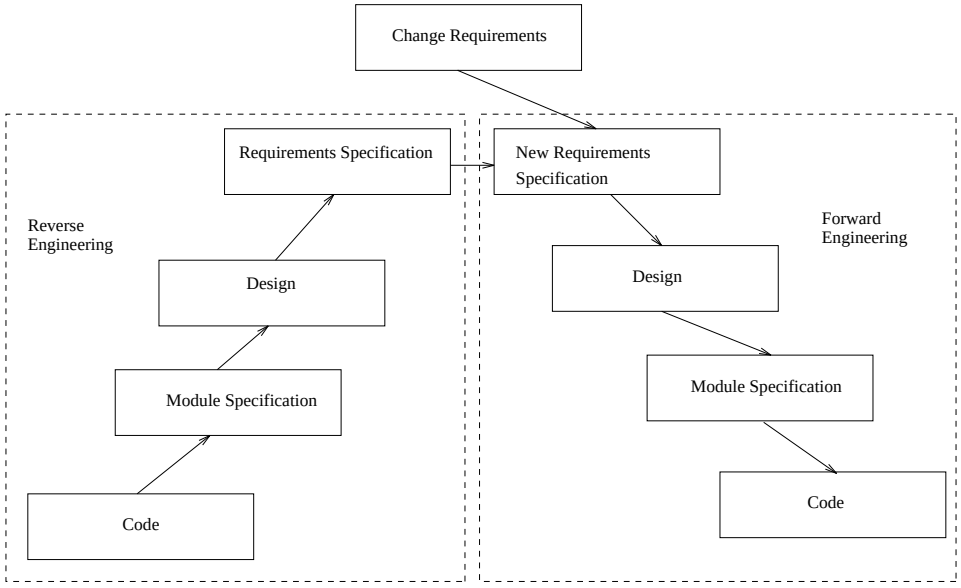

Figure 13.2: Maintenance process model 2

4. What do you mean by the term software reverse engineering? Why is it required? Explain the different activites undertaken during reverse engineering.
   Ans:
   Software reverse engineering is the process of recovering the design and the requirements specification of a product from an analysis of its code. The purpose of reverse engineering is to facilitate maintenance work by improving the understandability of a system and to produce the necessary documents for a legacy system.

   The different reverse engineering activities are summarized in Fig. 3.

5. What do you mean by the term software reengineering? Why is it required?
   Ans:
   The reengineering approach can be represented by a reverse engineering cycle followed by a forward engineering cycle. An important advantage of this approach is that it produces a more structured design compared to what the original product had, produces good documentation, and very often results in increased efficiency. The efficiency improvements are brought about by a more efficient design.

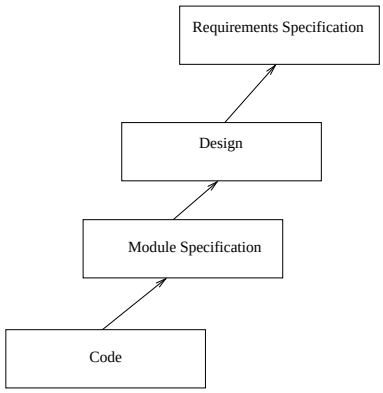

Figure 13.3: A process model for reverse engineering

6. If a software product costed Rs. 10,000,000/- for development, compute the annual maintenance cost given that every year approximately 5% of the code needs modification. Identify the factors which render the maintenance cost estimation inaccurate?
   Ans:

   Annual maintenance cost=10,000,000*5/100=Rs. 500,000

   The estimate is inaccurate as it does not take into account many important factors such as the experience level of the engineers carrying out maintenance, their familiarity with the product, etc.

7. What is a legacy software product? Explain the problems on would encounter while maintaining a legacy product.
   Ans:
   A legacy system as any software system that is hard to maintain. The typical problems associated with legacy systems are poor documentation, unstructured (spaghetti code with ugly control structure), and lack of personnel knowledgeable in the product. Many of the legacy systems were developed long time back. But, it is possible that a recently developed system having poor design and documentation can be considered to be a legacy system.

Chapter 14

Software Reuse

1. Why is it important for an organization to undertake an effective reuse program? What are the important reuse artifacts that can be reused? Why is reuse of software components much more difficult than hardware components?
   Ans:
   Software products are expensive. Software project managers are worried about the high cost of software development and are desperately looking for ways to cut development cost. A possible way to reduce development cost is to reuse parts from previously developed software. In addition to reduced development cost and time, reuse also leads to higher quality of the developed products since the reusable components are ensured to have high quality.

   Almost all artifacts associated with software development, including project plan and test plan can be reused. However, the prominent items that can be effectively reused are:

   - Requirements specification
   - Design
   - Code
   - Test cases
   - Knowledge

   Software components are more complicated than hardware components and the input and output are not standardized.

2. Identify the reasons why reuse of mathematical software is so successful. Also, identify the reasons why the reuse of software components

other than those of the mathematical software is difficult.
Ans:
Input and output to mathematical software are standardized.

3. What do you understand by the term *reuse domain*? Explain domain analysis and how domain analysis leads to increased component reusability.
Ans:
A reuse domain is a technically related set of application areas. A body of information is considered to be a problem domain for reuse, if a deep and comprehensive relationship exists among the information items as characterized by patterns of similarity among the development components of the software product. A reuse domain is a shared understanding of some community, characterized by concepts, techniques, and terminologies that show some coherence. Examples of domains are accounting software domain, banking software domain, business software domain, manufacturing automation software domain, telecommunication software domain, etc.

During domain analysis, a specific community of software developers get together to discuss community-wide solutions. Analysis of the application domain is required to identify the reusable components. The actual construction of the reusable components for a domain is called domain engineering.

4. Do you agree with the statement: "code" is the most important reuse artifact that can be used during software development"? Justify your answer.
Ans:

No. Knowledge is the most important reuse artifact that can be used during software development. A planned reuse of knowledge can increase the effectiveness of reuse. For this, the reusable knowledge should be systematically extracted and documented. But, it is usually very difficult to extract and document reusable knowledge.

5. What do you understand by the term "domain analysis"? How does domain analysis increase software reusability?
Ans:
A reuse domain is a technically related set of application areas. A body of information is considered to be a problem domain for reuse, if

a deep and comprehensive relationship exists among the information items as characterized by patterns of similarity among the development components of the software product. A reuse domain is a shared understanding of some community, characterized by concepts, techniques, and terminologies that show some coherence. Examples of domains are accounting software domain, banking software domain, business software domain, manufacturing automation software domain, telecommunication software domain, etc.

During domain analysis, a specific community of software developers get together to discuss community-wide solutions. Analysis of the application domain is required to identify the reusable components. The actual construction of the reusable components for a domain is called domain engineering.

6. Identify the stages through which a reuse domain progresses.
   Ans:
   The various stages through which a reuse domain progresses are the following: **Stage 1:** There is no clear and consistent set of notations. Obviously, no reusable components are available. All software is written from scratch.
   Stage 2: Here, only experience from similar projects are used in a development effort. This means that there is only knowledge reuse.
   Stage 3: At this stage, the domain is ripe for reuse. The set of concepts are stabilized and the notations standardized. Standard solutions to standard problems are available. There is both knowledge and component reuse.
   Stage 4: The domain has been fully explored. The software development for the domain can be largely be automated. Programs are not written in the traditional sense any more. Programs are written using a domain specific language, which is also known as an application generator.

7. Explain why reuse is difficult in software development compared to hardware development.
   Ans:

   Software components are more complicated than hardware components and the input and output are not standardized.

8. Explain why reuse of mathematical functions is easier compared to reuse of non-mathematical functions.
   Ans:
   Input and output to mathematical software are standardized.

9. What do you understand by the term "faceted classification" in the context of software reuse? How does faceted classification simplify component search in a component store?
   Ans:
   In faceted classification, each component is best described using a number of different characteristics or facets. For example, objects can be classified using the following:

   - actions they embody
   - objects they manipulate
   - data structures used
   - systems they are part of, etc.

   In a faceted classification scheme searching is more effective than a fully enumertive scheme since approximate searches can be made.

10. Explain how the faceted component classification (Prieto-Diaz's scheme) can be used for approximate searching. What are the advantages of approximate searching over exact searching?
    Ans:
    The approximate automated search locates products that appear to fulfill some of the specified requirements. The items located through the approximate search serve as a starting point for browsing the repository. These serve as the starting point for browsing the repository. The developer may follow links to other products until a sufficiently good match is found.

11. Suppose your team has developed a software product. How would you assess the potential reusability of the developed functions?
    Ans:
    Assessment of a components reuse potential can be obtained from an analysis of a questionnaire circulated among the developers. The questionnaire can be devised to assess a component's reusability. The programmers working in similar application domain can be used to answer the questionnaire about the product's reusability. Depending on the answers given by the programmers, either the component be taken up for reuse as it is, it is modified and refined before it is entered

into the reuse repository, or it is ignored. A sample questionnaire to assess a component's reusability is the following.

- Is the component's functionality required for implementation of systems in the future?
- How common is the component's function within its domain?
- Would there be a duplication of functions within the domain if the component is taken up?
- Is the component hardware dependent?
- Is the design of the component optimized enough?
- If the component is non-reusable, then can it be decomposed to yield some reusable components?
- Can we parametrize a non-reusable component so that it becomes reusable?

12. In an organization level software reuse, identify aspects of a developed function which hamper its reusability. How can you improve reusability of the components you have identified for reuse?
Ans:
Extracting reusable components from projects that were completed in the past presents an important difficulty not encountered while extracting a reusable component from an ongoing project — typically, the original developers are no longer available for consultation. Development of new systems leads to an assortment of products, since reusability ranges from items whose reusability is immediate to those items whose reusability is highly improbable.

    Achieving organization-level reuse requires adoption of the following steps:

    - assess of an item's potential for reuse
    - refine the item for greater reusability
    - enter the product in the reuse repository

13. What is an application generator? Why reuse is easier while using an application generator compared to a component library? What are the shortcomings of an application generator?
Ans:
Application generators translate specifications into application programs. The specification usually is written using 4GL. The specification might also in a visual form. The programmer would create a

graphical drawing using some standard available symbols. Defining what is variant and what is invariant corresponds to parameterizing a subroutine to make it reusable. A subroutine's parameters are variants because the programmer can specify them while calling the subroutine. Parts of a subroutine that are not parameterized, cannot be changed.

Application generators have significant advantages over simple parameterized programs. The biggest of these is that the application generators can express the variant information in an appropriate language rather than being restricted to function parameters, named constants, or tables. The other advantages include fewer errors, easier to maintain, substantially reduced development effort, and the fact that one need not bother about the implementation details. Application generators are handicapped when it is necessary to support some new concepts or features. Some application generators overcome this handicap through an escape mechanism. Programmers can write code in some 3GL through this mechanism.

14. Devise a scheme to store software reuse artifacts. Explain how components can be searched in your scheme.