EXPERIMENT-3

AIM: ER Modelling From the Problem Statements.

Entity Relationship Model

Entity-Relationship model is used to represent a logical design of a database to be created. In ER model, real world objects (or concepts) are abstracted as entities, and different possible associations among them are modeled as relationships.For example, student and school -- they are two entities. Students study in school. So, these two entities are associated with a relationship "Studies in".As another example, consider a system where some job runs every night, which updates the database. Here, job and database could be two entities. They are associated with the relationship "Updates".

Entity Set and Relationship Set

An entity set is a collection of all similar entities. For example, "Student" is an entity set that abstracts all students. Ram, John are specific entities belonging to this set. Similarly, a "Relationship" set is a set of similar relationships.

Attributes of Entity

Attributes are the characteristics describing any entity belonging to an entity set. Any entity in a set can be described by zero or more attributes.For example, any student has got a name, age, an address. At any given time a student can study only at one school. In the school he would have a roll number, and of course a grade in which he studies. These data are the attributes of the entity set Student.

Keys

One or more attribute(s) of an entity set can be used to define the following keys:

- **Super key:** One or more attributes, which when taken together, helps to uniquely identify an entity in an entity set. For example, a school can have any number of students. However, if we know grade and roll number, then we can uniquely identify a student in that school.

- **Candidate key:** It is a minimal subset of a super key. In other words, a super key might contain extraneous attributes, which do not help in identifying an object uniquely. When such attributes are removed, the key formed so is called a candidate key.

- **Primary key:** A database might have more than one candidate key. Any candidate key chosen for a particular implementation of the database is called a primary key.

- **Prime attribute:** Any attribute taking part in a super key

Weak Entity

An entity set is said to be weak if it is dependent upon another entity set. A weak entity can't be uniquely identified only by it's attributes. In other words, it doesn't have a super key.For example, consider a company that allows employees to have travel allowance for their immediate family. So, here we have two entity sets: employee and family, related by "Can claim for". However, family doesn't have a super key. Existence of a family is entirely dependent on the concerned employee. So, it is meaningful only with reference to employee.

Entity Generalization and Specialization

Once we have identified the entity sets, we might find some similarities among them. For example, multiple person interacts with a banking system. Most of them are customers, and rest employees or other service providers. Here, customers, employees are persons, but with certain specializations. Or in other way, person is the generalized form of customer and employee entity sets.ER model uses the "ISA" hierarchy to depict specialization (and thus, generalization).

Mapping Cardinalities

One of the main tasks of ER modeling is to associate different entity sets. Let's consider two entity sets E1 and E2 associated by a relationship set R. Based on the number of entities in E1 and E2 are associated with, we can have the following four type of mappings:

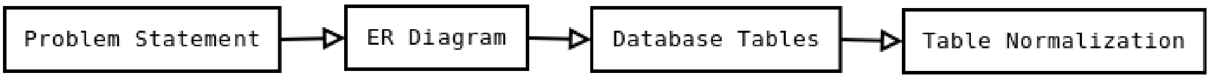
- **One to one:** An entity in E1 is related to at most a single entity in E2, and vice versa
- **One to many:** An entity in E1 could be related to zero or more entities in E2. Any entity in E2 could be related to at most a single entity in E1.
- **Many to one:** Zero or more number of entities in E1 could be associated to a single entity in E2. However, an entity in E2 could be related to at most one entity in E1.
- **Many to many:** Any number of entities could be related to any number of entities in E2, including zero, and vice versa.

ER Diagram

From a given problem statement we identify the possible entity sets, their attributes, and relationships among different entity sets. Once we have these information, we represent them pictorially, called an entity-relationship (ER) diagram.

Importance of ER modeling

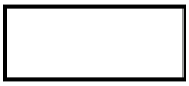
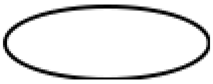
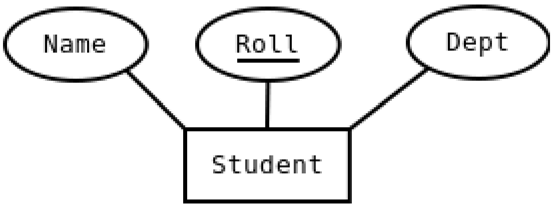
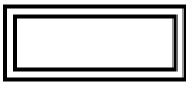
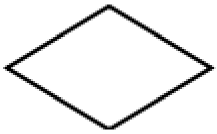
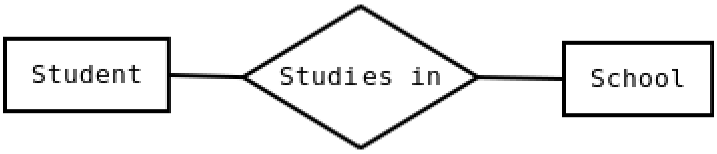
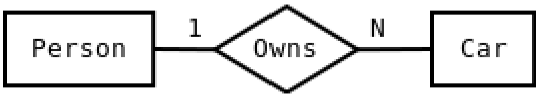
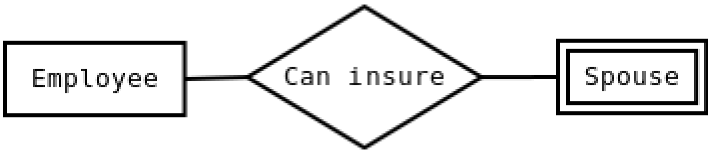
Figure - 01 shows the different steps involved in implementation of a (relational) database.



Database design steps Figure - 01: Steps to implement a RDBMS

Given a problem statement, the first step is to identify the entities, attributes and relationships. We represent them using an ER diagram. Using this ER diagram, table structures are created, along with required constraints. Finally, these tables are normalized in order to remove redundancy and maintain data integrity. Thus, to have data stored efficiently, the ER diagram is to be drawn as much detailed and accurate as possible.

Graphical Notations for ER Diagram

Term	Notation	Remarks
Entity set		Name of the set is written inside the rectangle
Attribute		Name of the attribute is written inside the ellipse
Entity with attributes		Roll is the primary key; denoted with an underline
Weak entity set		
Relationship set		Name of the relationship is written inside the diamond
Related enity sets		
Relationship cardinality		A person can own zero or more cars but no two persons can own the same car
Relationship with weak entity set		

CASE STUDY

The first step towards ER modelling is to identify the set of relevant entities from the given problem statement. The two primary, and obvious, entity sets in this context are "Member" and "Book". The entity set "Member" represents all students, professors, or employees who have registered themselves with the LIS. While registering with the LIS one has to furnish a lot of personal and professional information. This typically includes name (well, that is trivial), employee ID (roll # for students), email address, phone #, age, date of joining in this institute. The system may store some not-so-important information as well like, blood group, marital status, and so on. All these pieces of information that an user has to provide are sufficient to describe a particular member. These characteristics are the attributes of the entities belonging to the entity set "Member". It is essential for an entity to have one or more attributes that help us to distinguish it from another entity. 'Name' can't help that -- two persons could have exactly the same name. However, ('Name', 'Phone #') combination seems to be okay. No two persons can have the same phone number. 'Employee ID', 'Email address' are other potential candidates. Here, 'Employee ID', 'Email address' and ('Name', 'Phone #') are super keys. We choose 'Employee ID' to uniquely identify an user in our implementation. So, 'Employee ID' becomes our primary key (PK) for the "Member" entity set.

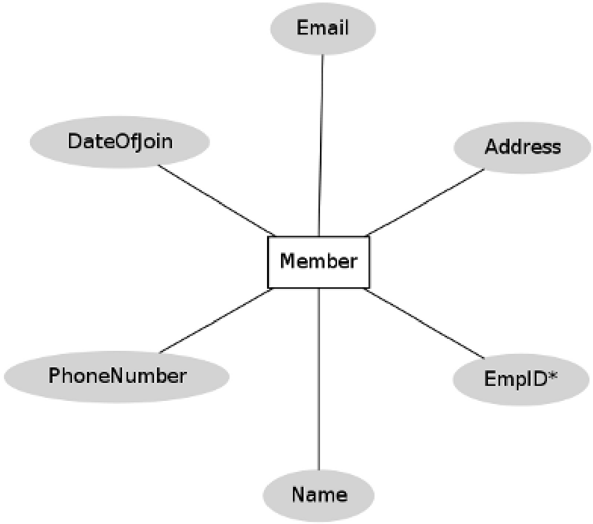


Figure 1: "Member" entity set

A graphical representation of the "Book" entity set is shown in figure 2

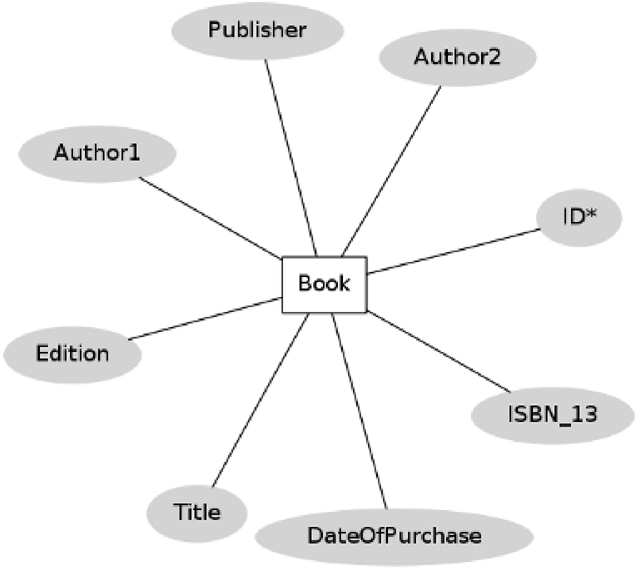


Figure 2: "Book" entity set

EXPERIMENT-4

AIM: State chart and Activity Modelling.

Statechart diagrams

In case of Object Oriented Analysis and Design, a system is often abstracted by one or more classes with some well defined behaviour and states. A statechart diagram is a pictorial representation of such a system, with all it's states, and different events that lead transition from one state to another.

Statechart diagrams are normally drawn to model the behaviour of a complex system. For simple systems this is optional.

Building Blocks of a Statechart Diagram:

State: A state is any "distinct" stage that an object (system) passes through in it's lifetime. An object remains in a given state for finite time until "something" happens, which makes it to move to another state

Transition: It is movement from one state to another state in response to an external stimulus (or any internal event). A transition is represented by a solid arrow from the current state to the next state.

Action: As mentioned in [ii], actions represents behaviour of the system. While the system is performing any action for the current event, it doesn't accept or process any new event.

Guidelines for drawing Statechart Diagrams

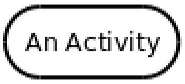
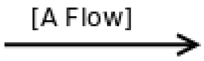
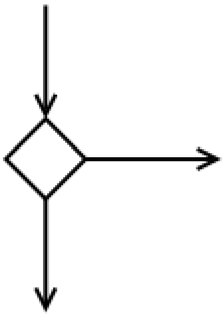
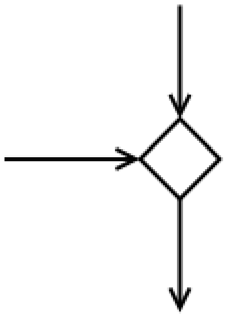
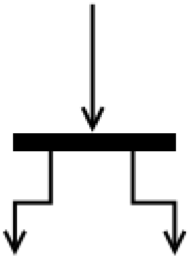
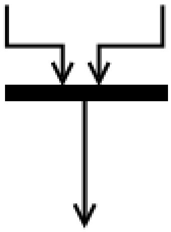
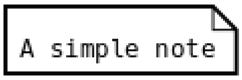
Following steps could be followed, as suggested in todraw a statechart diagram:

- For the system to developed, identify the distinct states that it passes through
- Identify the events (and any precondition) that cause the state transitions. Often these would be the methods of a class as identified in a class diagram.
- Identify what activities are performed while the system remains in a given state

Activity Diagrams

Activity diagrams fall under the category of behavioural diagrams in Unified Modeling Language. It is a high level diagram used to visually represent the flow of control in a system. It has similarities with traditional flow charts. However, it is more powerful than a simple flow chart since it can represent various other concepts like concurrent activities, their joining, and so on . Activity diagrams, however, cannot depict the message passing among related objects. As such, it can't be directly translated into code. These kind of diagrams are suitable for confirming the logic to be implemented with the business users. These diagrams are typically used when the business logic is complex. In simple scenarios it can be avoided entirely

The following table shows commonly used components with a typical activity diagram:

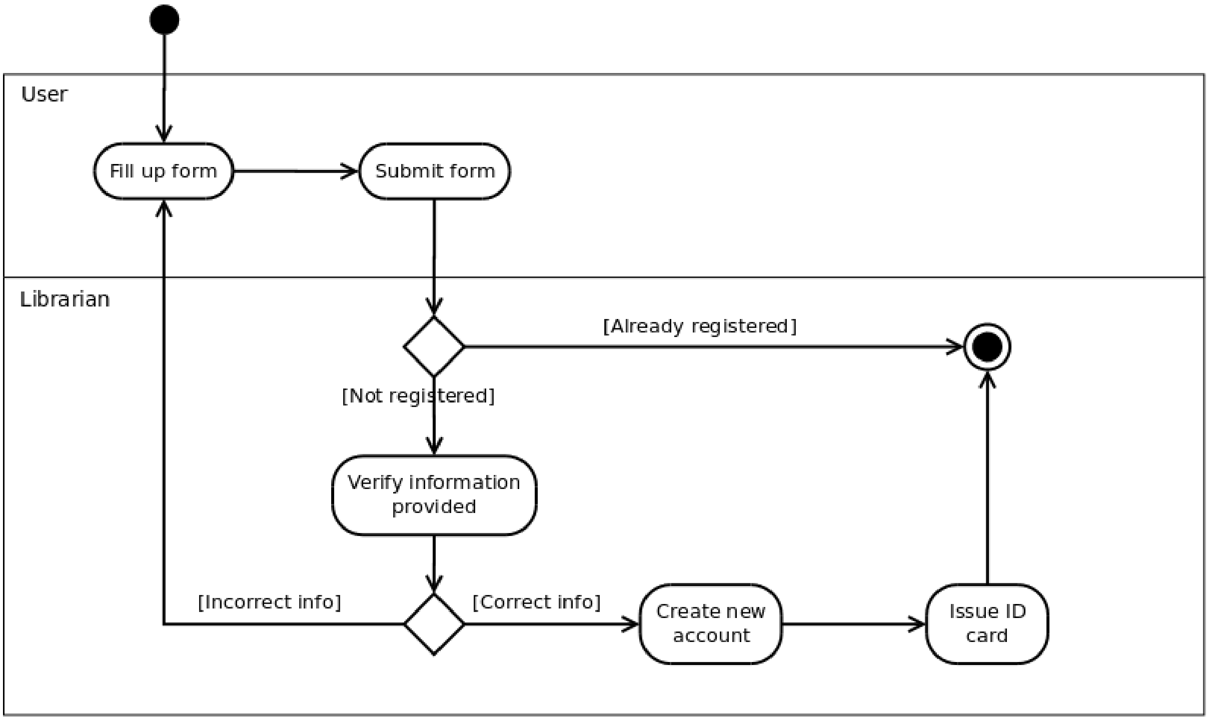
Component	Graphical Notation
Activity	
Flow	
Decision	
Merge	
Fork	
Join	
Note	

The SE VLabs Institute has been recently setup to provide state-of-the-art research facilities in the field of Software Engineering. Apart from research scholars (students) and professors, it also includes quite a large number of employees who work on different projects undertaken by the institution.

As the size and capacity of the institute is increasing with the time, it has been proposed to develop a Library Information System (LIS) for the benefit of students and employees of the institute. LIS will enable the members to borrow a book (or return it) with ease while sitting at his desk/chamber. The system also enables a member to extend the date of his borrowing if no other booking for that particular book has been made. For the library staff, this system aids them to easily handle day-to-day book transactions. The librarian, who has administrative privileges and complete control over the system, can enter a new record into the system when a new book has been purchased, or remove a record in case any book is taken off the shelf. Any non-member is free to use this system to browse/search books online. However, issuing or returning books is restricted to valid users (members) of LIS only.

From the given problem we can identify at least four different functionality offered by the system:

Register a new member
Issue book
Reissue book
Update inventory
To begin with, let's consider the activity diagram for user registration, as shown in figure - 01.



User Registration Figure-01: Activity diagram for new user registration
A new user fills up the registration form for library membership (either online or in paper), and submits to the librarian. Of course, an already registered user can't create another account for himself (or, herself). For users' who don't have an account already and have submitted their registration forms, the librarian verifies the information provided, possibly against the central database used by the institution. If all information have been provided correctly, librarian goes on with creating a new

account for the user. Otherwise, the user is asked to provide all and correct information in his (her) registration form. Once a new account has been created for the user, he (she) is being issued an ID card, which is to be provided for any future transaction in the library.

Note that in the above diagram two swim lanes haven been shown indicated by the labels User and Librarian. The activities have been placed in swim lanes that correspond to the relevant role.

One of the major events that occur in any library is issue of books to it's members. Figure-02 tries to depict the workflow involved while issuing books.

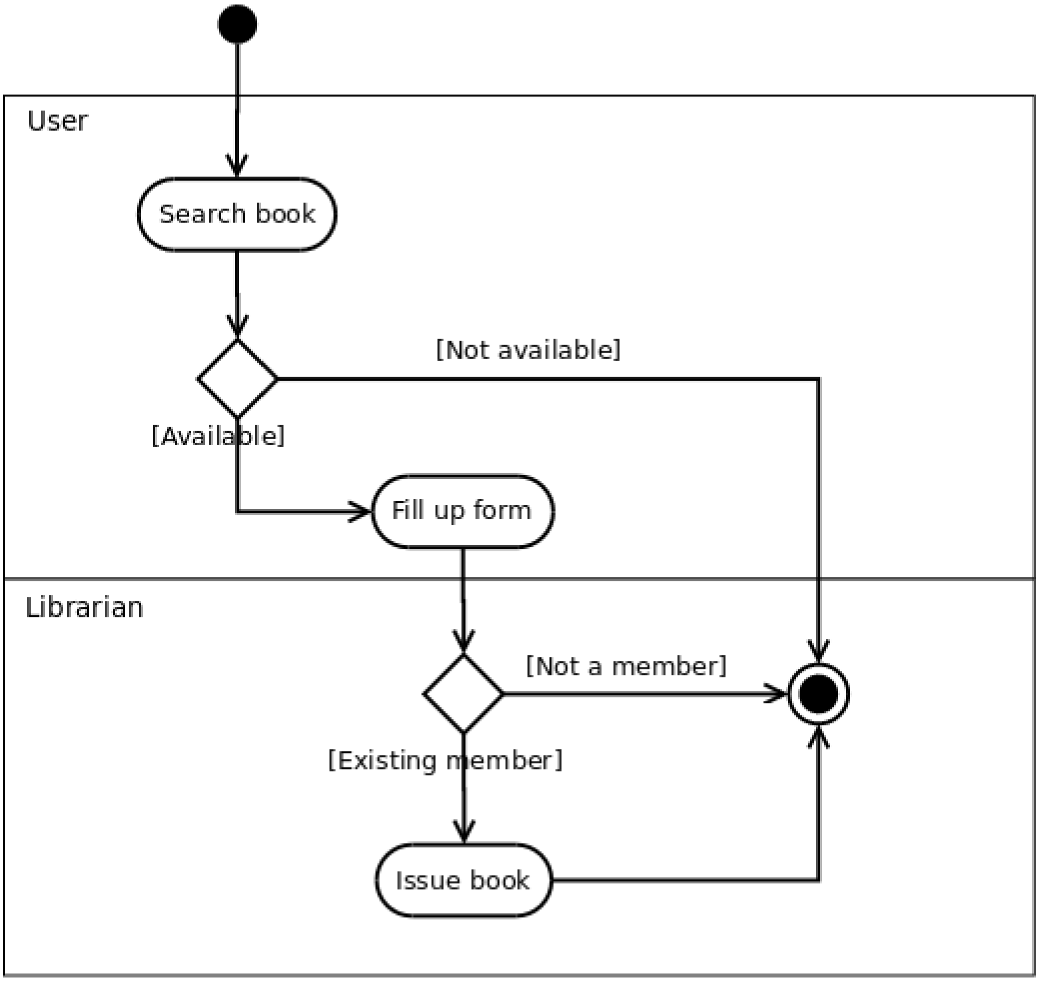


Figure-02:Activity diagram for issuing books

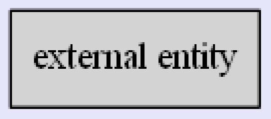
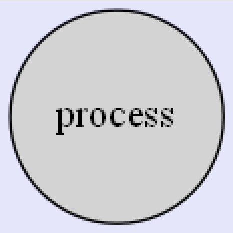
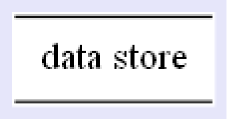
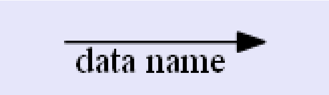
EXPERIMENT-5

AIM: Modelling Data Flow Diagrams.

Data Flow Diagram

DFD provides the functional overview of a system. The graphical representation easily overcomes any gap between ’user and system analyst’ and ‘analyst and system designer’ in understanding a system. Starting from an overview of the system it explores detailed design of a system through a hierarchy. DFD shows the external entities from which data flows into the process and also the other flows of data within a system. It also includes the transformations of data flow by the process and the data stores to read or write a data.

Graphical notations for Data Flow Diagram

Term	Notation	Remarks
External entity		Name of the external entity is written inside the rectangle
Process		Name of the process is written inside the circle
Data store		A left-right open rectangle is denoted as data store; name of the data store is written inside the shape
Data flow		Data flow is represented by a directed arc with its data name

Explanation of Symbols used in DFD

Process: Processes are represented by circle. The name of the process is written into the circle. The name of the process is usually given in such a way that represents the functionality of the process. More detailed functionalities can be shown in the next Level if it is required. Usually it is better to keep the number of processes less than 7 If we see that the number of processes becomes more than 7 then we should combine some the processes to a single one to reduce the number of processes and further decompose it to the next level .

External entity: External entities are only appear in context diagram.External entities are represented by a rectangle and the name of the external entity is written into the shape. These send data to be processed and again receive the processed data.

Data store: Data stares are represented by a left-right open rectangle. Name of the data store is written in between two horizontal lines of the open rectangle. Data stores are used as repositories from which data can be flown in or flown out to or from a process.

Data flow: Data flows are shown as a directed edge between two components of a Data Flow Diagram. Data can flow from external entity to process, data store to process, in between two processes and vice-versa.

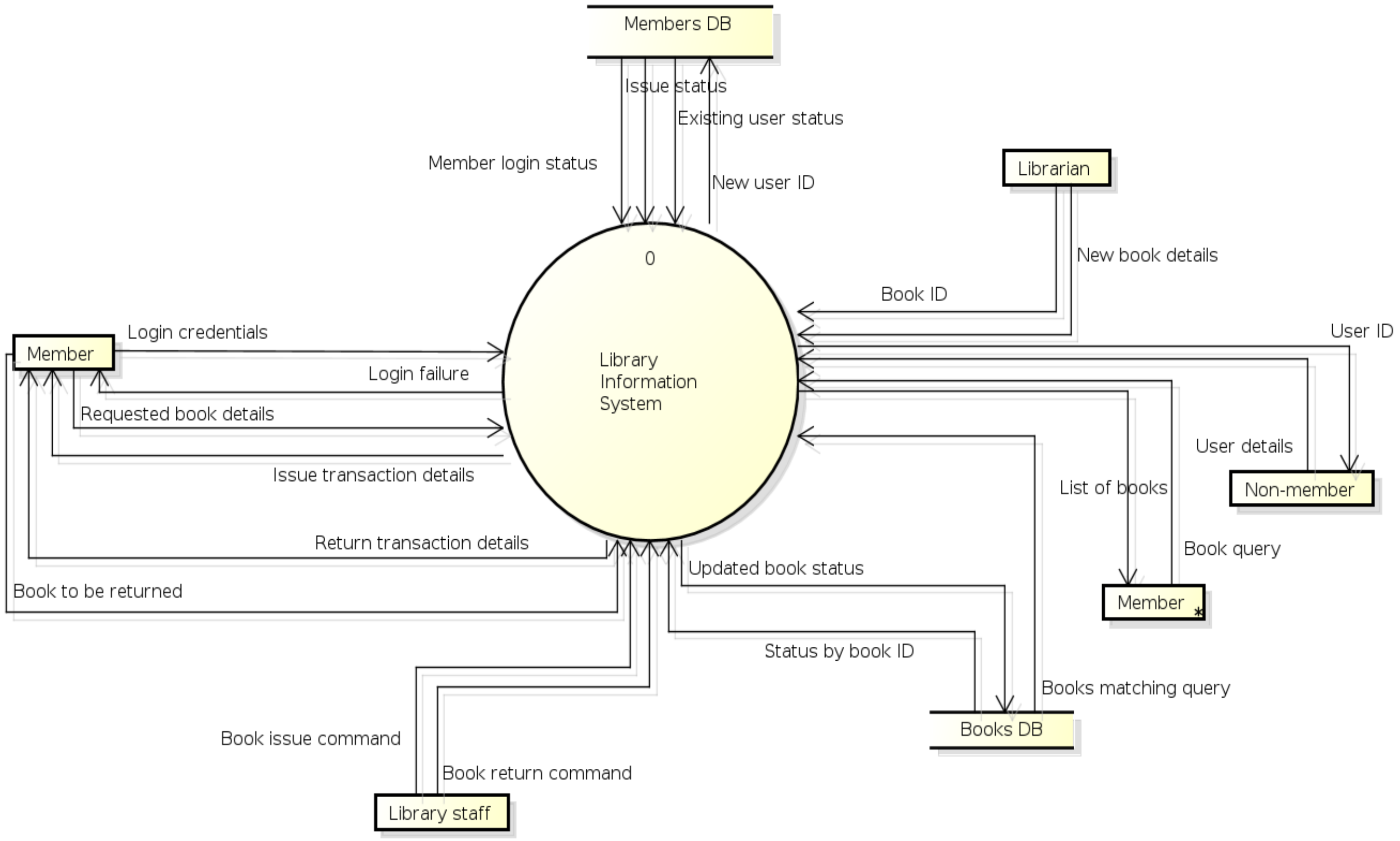
Context diagram and leveling DFD

We start with a broad overview of a system represented in level 0 diagram. It is known as context diagram of the system. The entire system is shown as single process and also the interactions of external entities with the system are represented in context diagram.Further we split the process in next levels into several numbers of processes to represent the detailed functionalities performed by the system. Data stores may appear in higher level DFDs.Numbering of processes : If process ‘p’ in context diagram is split into 3 processes ‘p1’, ‘p2’and ‘p3’ in next level then these are labeled as 0.1, 0.2 and 0.3 in level 1 respectively. Let the process ‘p3’ is again split into three processes ‘p31’, ‘p32’ and ‘p33’ in level 2, so, these are labeled as 0.3.1, 0.3.2 and 0.3.3 respectively and so on.Balancing DFD: The data that flow into the process and the data that flow out to the process need to be match when the process is split into in the next level[2]. This is known as balancing a DFD.

CASE STUDY

The SE VLabs Institute has been recently setup to provide state-of-the-art research facilities in the field of Software Engineering. Apart from research scholars (students) and professors, it also includes quite a large number of employees who work on different projects undertaken by the institution.As the size and capacity of the institute is increasing with the time, it has been proposed to develop a Library Information System (LIS) for the benefit of students and employees of the institute. LIS will enable the members to borrow a book (or return it) with ease while sitting at his desk/chamber. The system also enables a member to extend the date of his borrowing if no other booking for that particular book has been made. For the library staff, this system aids them to easily handle day-to-day book transactions. The librarian, who has administrative privileges and complete control over the system, can enter a new record into the system when a new book has been purchased, or remove a record in case any book is taken off the shelf. Any non-member is free to use this system to browse/search books online. However, issuing or returning books is restricted to valid users (members) of LIS only.The final deliverable would a web application, which should run only within the institute LAN. Although this reduces security risk of the software to a large extent, care should be taken no confidential information (eg., passwords) is stored in plain text.

Figure 1 shows the context-level DFD for LIS. The entire system is represented with a single circle (process). The external entities interacting with this system are members of LIS, library staff, librarian, and non-members of LIS. Two database are used to keep track of member information and details of books in the library.Let us focus on the external entity, Member. In order to issue or return books a member has to login to the system. The data flow labeled with "Login credentials" indicate the step when a member authenticates himself by providing required information (user ID, password). The system in turn verifies the user credentials using information stored in the members database. If all information are not provided correctly, the user is shown a login failure message. Otherwise, the user can continue with his operation. Note that a DFD does not show conditional flows. It can only summarize the information flowing in and out of the system.The data flow with the label "Requested book details" identify the information that the user has to provide in order to issue a book. LIS checks with the books database whether the given book is available. After a book has been issued, the transaction details is provided to the member

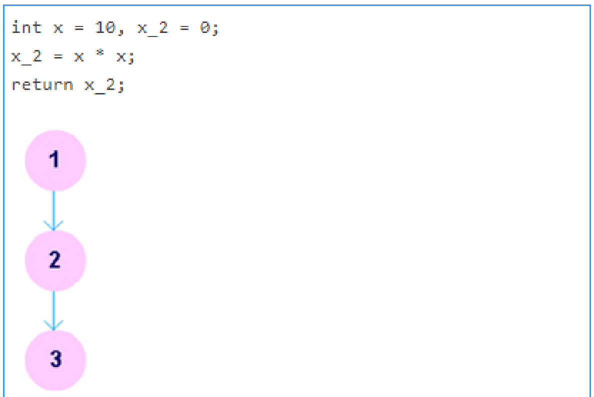


EXPERIMENT-6

AIM: Estimation of Test Coverage Metrics and Structural Complexity.

Control Flow Graph

A control flow graph (CFG) is a directed graph where the nodes represent different instructions of a program, and the edges define the sequence of execution of such instructions. Figure 1 shows a small snippet of code (compute the square of an integer) along with it's CFG. For simplicity, each node in the CFG has been labeled with the line numbers of the program containing the instructions. A directed edge from node #1 to node #2 in figure 1 implies that after execution of the first statement, the control of execution is transferred to the second instruction.



```
int x = 10, x_2 = 0;
x_2 = x * x;
return x_2;
```

Figure 1: A simple program and it's CFG

A program, however, doesn't always consist of only sequential statements. There could be branching and looping involved in it as well. Figure 2 shows how a CFG would look like if there are sequential, selection and iteration kind of statements in order.

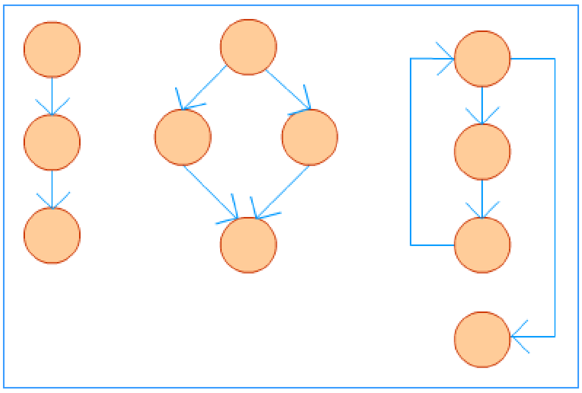


Figure 2: CFG for different types of statements

Terminologies

Path

A path in a CFG is a sequence of nodes and edges that starts from the initial node (or entry block) and ends at the terminal node. The CFG of a program could have more than one terminal nodes.

Linearly Independent Path

A linearly independent path is any path in the CFG of a program such that it includes at least one new edge not present in any other linearly independent path. A set of linearly independent paths give a clear picture of all possible paths that a program can take during it's execution. Therefore, path-coverage testing of a program would suffice by considering only the linearly independent paths.
In figure 3 we can find four linearly independent paths:

1 - 3 - 6 - (7, 8) - 10
1 - 3 - 6 - (7, 8) - 9 - 10
1 - 3 - (4, 5) - 6 - (7, 8) - 10
1 - 3 - (4, 5) - 6 - (7, 8) - 9 - 10

Note that 1 - 3 - (4, 5) - 3 - (4, 5) - 6 - (7, 8) - 10, for instance, won't qualify as a linearly independent path because there is no new edge not already present in any of the above four linearly independent paths.

McCabe's Cyclomatic Complexity

McCabe had applied graph-theoretic analysis to determine the complexity of a program module [vi]. Cyclomatic complexity metric, as proposed by McCabe, provides an upper bound for the number of linearly independent paths that could exist through a given program module. Complexity of a module increases as the number of such paths in the module increase. Thus, if Cyclomatic complexity of any program module is 7, there could be up to seven linearly independent paths in the module. For a complete testing, each of those possible paths should be tested.

Optimum Value of Cyclomatic Complexity

A set of threshold values for Cyclomatic complexity has been presented in , which we reproduce below.

V(G)	Module Category	Risk
1-10	Simple	Low
11-20	More complex	Moderate
21-50	Complex	High
> 50	Unstable	Very high

CASE STUDY

The SE VLabs Institute has been recently setup to provide state-of-the-art research facilities in the field of Software Engineering. Apart from research scholars (students) and professors, it also includes quite a large number of employees who work on different projects undertaken by the institution.As the size and capacity of the institute is increasing with the time, it has been proposed to develop a Library Information System (LIS) for the benefit of students and employees of the institute. LIS will enable the members to borrow a book (or return it) with ease while sitting at his desk/chamber. The system also enables a member to extend the date of his borrowing if no other booking for that particular book has been made. For the library staff, this system aids them to easily handle day-to-day book transactions. The librarian, who has administrative privileges and complete control over the system, can enter a new record into the system when a new book has been purchased, or remove a record in case any book is taken off the shelf. Any non-member is free to use this system to browse/search books online. However, issuing or returning books is restricted to valid users (members) of LIS only.The final deliverable would a web application (using the recent HTML 5), which should run only within the institute LAN. Although this reduces security risk of the software to a large extent, care should be taken no confidential information (eg., passwords) is stored in plain text.

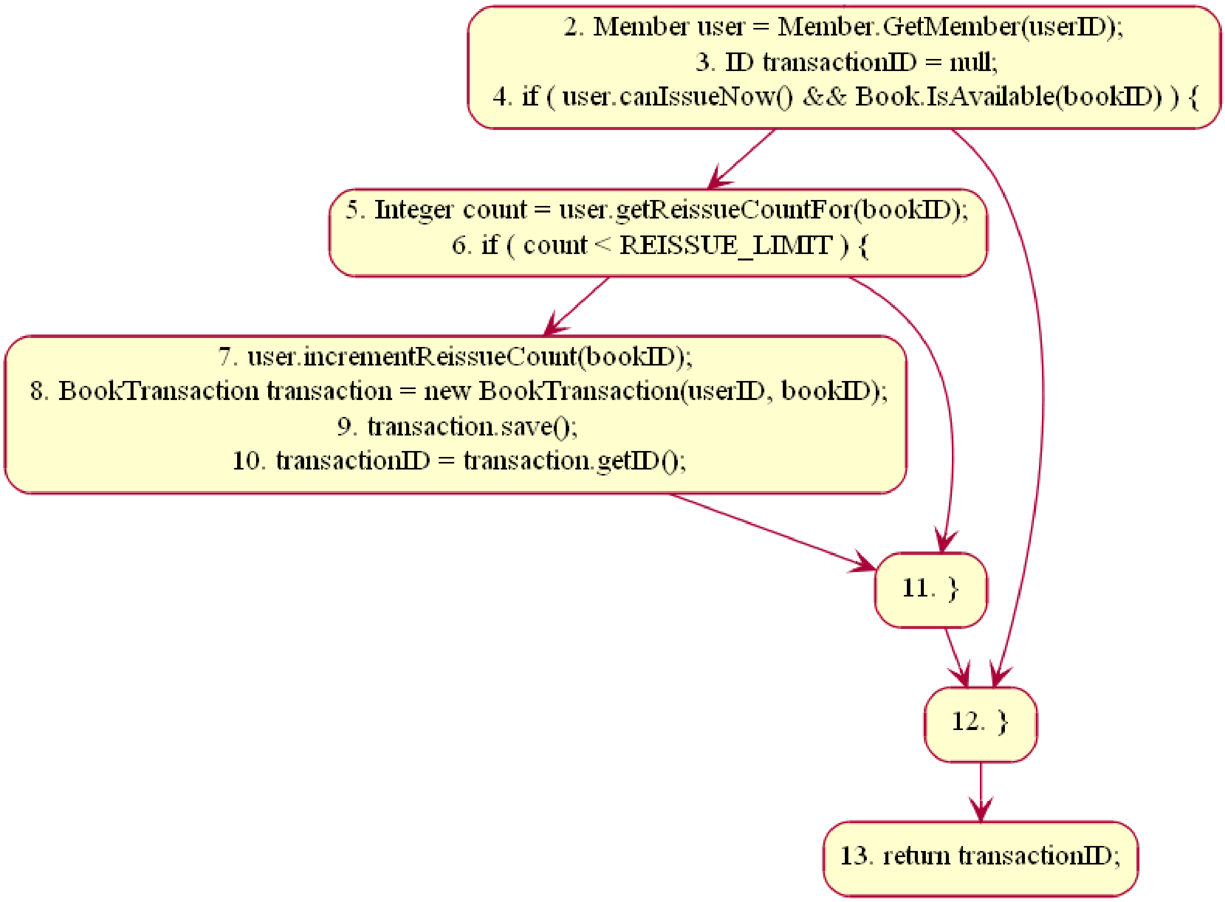

2. Member user = Member.GetMember(userID);
3. ID transactionID = null;
4. if (user.canIssueNow() && Book.IsAvailable(bookID)) {
5. Integer count = user.getReissueCountFor(bookID);
6. if (count < REISSUE_LIMIT) {
7. user.incrementReissueCount(bookID);
8. BookTransaction transaction = new BookTransaction(userID, bookID);
9. transaction.save();
10. transactionID = transaction.getID();
11. }
12. }
13. return transactionID;

Figure 1. CFG for "ReissueBook" method

EXPERIMENT-7

AIM: Designing Test Suites.

Software Testing

Testing software is an important part of the development life cycle of a software. It is an expensive activity. Hence, appropriate testing methods are necessary for ensuring the reliability of a program. According to the ANSI/IEEE 1059 standard, the definition of testing is the process of analyzing a software item, to detect the differences between existing and required conditions i.e. defects/errors/bugs and to evaluate the features of the software item.The purpose of testing is to verify and validate a software and to find the defects present in a software. The purpose of finding those problems is to get them fixed.

- Verification is the checking or we can say the testing of software for consistency and conformance by evaluating the results against pre-specified requirements.
- Validation looks at the systems correctness, i.e. the process of checking that what has been specified is what the user actually wanted.
- Defect is a variance between the expected and actual result. The defect's ultimate source may be traced to a fault introduced in the specification, design, or development (coding) phases.

Types of Software Testing

Testing is done in every stage of software development life cycle, but the testing done at each level of software development is different in nature and has different objectives. There are different types of testing, such as stress testing, volume testing, configuration testing, compatibility testing, recovery testing, maintenance testing, documentation testing, and usability testing. Software testing are mainlyof following types [1]

Unit Testing

Unit testing is done at the lowest level. It tests the basic unit of software, that is the smallest testable piece of software. The individual component or unit of a program are tested in unit testing. Unit testing are of two types.

Black box testing: This is also known as functional testing , where the test cases are designed based on input output values only. There are many types of Black Box Testing but following are the prominent ones.

- Equivalence class partitioning: In this approach, the domain of input values to a program is divided into a set of equivalence classes. e.g. Consider a software program that computes whether an integer number is even or not that is in the range of 0 to 10. Determine the equivalence class test suite. There are three equivalence classes for this program. - The set of negative integer - The integers in the range 0 to 10 - The integer larger than 10

- Boundary value analysis : In this approach, while designing the test cases, the values at boundaries of different equivalence classes are taken into consideration. e.g. In the above given example as in equivalence class partitioning, a boundary values based test suite is { 0, -1, 10, 11 }

White box testing: It is also known as structural testing. In this testing, test cases are designed on the basis of examination of the code.This testing is performed based on the knowledge of how the system is implemented. It includes analyzing data flow, control flow, information flow, coding practices, exception and error handling within the system, to test the intended and unintended software behavior. White box testing can be performed to validate whether code implementation.

System Testing

System testing tends to affirm the end-to-end quality of the entire system. System testing is often based on the functional / requirement specification of the system. Non-functional quality attributes, such as reliability, security, and maintainability are also checked. There are three types of system testing.

CASE STUDY

The SE VLabs Institute has been recently setup to provide state-of-the-art research facilities in the field of Software Engineering. Apart from research scholars (students) and professors, it also includes quite a large number of employees who work on different projects undertaken by the institution.

As the size and capacity of the institute is increasing with the time, it has been proposed to develop a Library Information System (LIS) for the benefit of students and employees of the institute. LIS will enable the members to borrow a book (or return it) with ease while sitting at his desk/chamber. The system also enables a member to extend the date of his borrowing if no other booking for that particular book has been made. For the library staff, this system aids them to easily handle day-to-day book transactions. The librarian, who has administrative privileges and complete control over the system, can enter a new record into the system when a new book has been purchased, or remove a record in case any book is taken off the shelf. Any non-member is free to use this system to browse/search books online. However, issuing or returning books is restricted to valid users (members) of LIS only.

The final deliverable would a web application (using the recent HTML 5), which should run only within the institute LAN. Although this reduces security risk of the software to a large extent, care should be taken no confidential information (eg., passwords) is stored in plain text.

As already discussed under the theory section, test case preparation could begin right after requirements identification stage. It is desirable (and advisable) to create a Requirements Traceability Matrix (RTM) showing a mapping from individual requirement to test case(s). A simplified form of the RTM is shown in table 1 (the numbers shown in this table are arbitrary; not specific to LIS).

Table 1: A simplified mapping from requirements to test cases

Requirement #	Test Case #
R1	TC1
R2	TC2, TC3, TC4
R3	TC5
R4	TC6