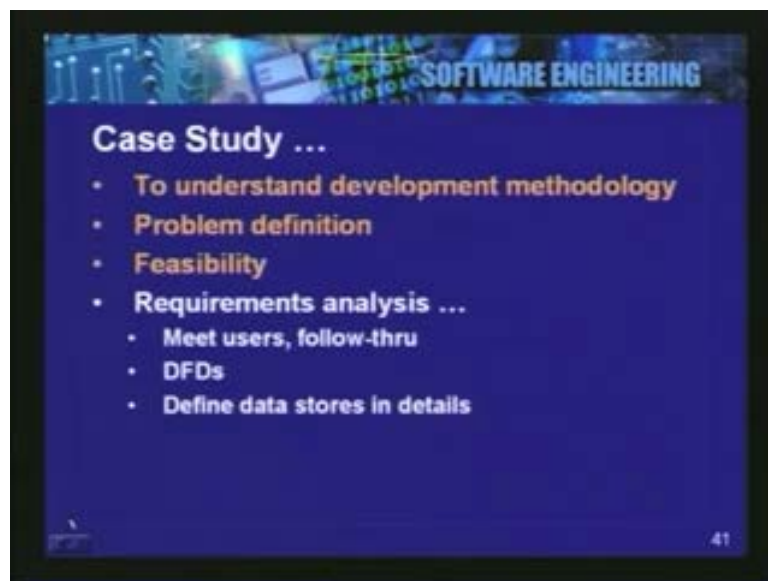


Software Engineering
Prof. N.L. Sarda
Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Lecture - 24
Case Study (Part – II)

Let us continue with the discussion of the case study which we started last time; this is about the library organization and we are taking a case study of a circulation function. The purpose of the case study is to understand the development methodology, the various phases in development, the kind of techniques that you use during carrying out these phases and also the deliverables.

We have seen the problem definition and we produced a problem definition document, we have done the feasibility, we have prepared two alternatives for the user with different costs and different benefits and we presented the feasibility report and we assume that one of the two alternatives were accepted by the user or the library in this case and we proceed to the important phase of requirement analysis.

(Refer Slide Time: 01:52 min)



In the requirement analysis we presented the information that we obtained from different users of the library. These included not only the members who borrow books from the library but also the counter clerks and library management and the idea was to understand the kind of work they do and relate them to the problems which are faced by the users so that we can suggest a more efficient and cost effective alternative. And having done that now we want to study the functions, the tasks and understand them in more depth and prepare a requirement analysis document where we will address ways by which these problems will be solved by the proposed system. So we met the users, we followed through some of the activities they performed and we prepared the dataflow diagrams and identified the different sub-processes and the different data stores. After that, we defined the contents of each data store. Unless we do this we will not be able to clearly understand what needs to be done in each of the sub-processes identified in the dataflow diagram.

So, if you recall, last time we had studied the contents of the data store called book which is one of the core data store which contains data about all the books available in the library.

(Refer Slide Time: 03:42 min)

SOFTWARE ENGINEERING

Requirements Analysis ...

- **contents of data store 'book'**

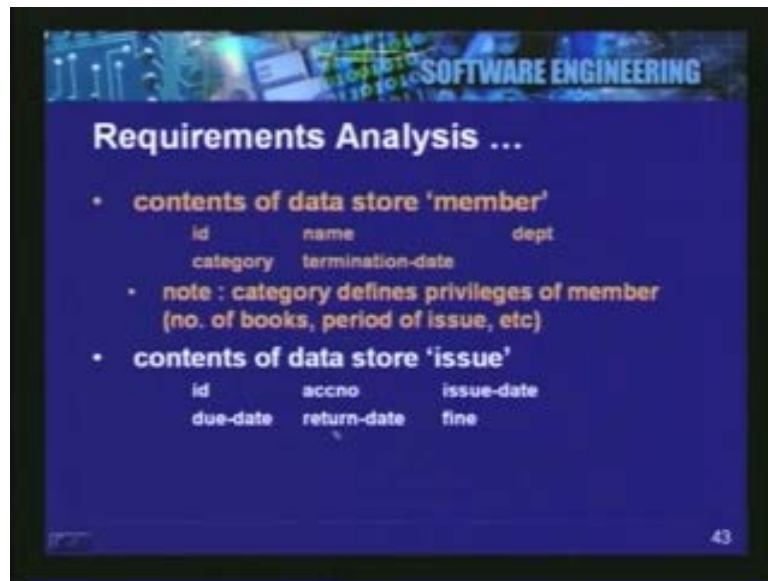
accno	ISBN	title
authors	publisher	year
price	classification	book-type
- **note : book-type indicates book category that may have issue restrictions (only to certain member categories and for certain periods)**

42

We then noted that accession number is an important key attribute for this data store and the book type is also an important attribute because book type will decide what kind of users can borrow this book. And then there are other attributes which are useful for enquiry purpose. So we have defined here the contents of data store book. this gives us a very clear idea of what kind of information is available in that data store so that the various processes which either obtain data from this data store or update something in this data store we can clearly understand them with reference to the contents here.

Let us now look at the other data stores quickly. We have the member data stores; these are the members (Refer Slide Time: 4:20) who are permitted to borrow books, every member has a identifier, name, a department, his category; and now the category could be faculty, student and so on and the date on which the membership would expire.

(Refer Slide Time: 04:39 min)

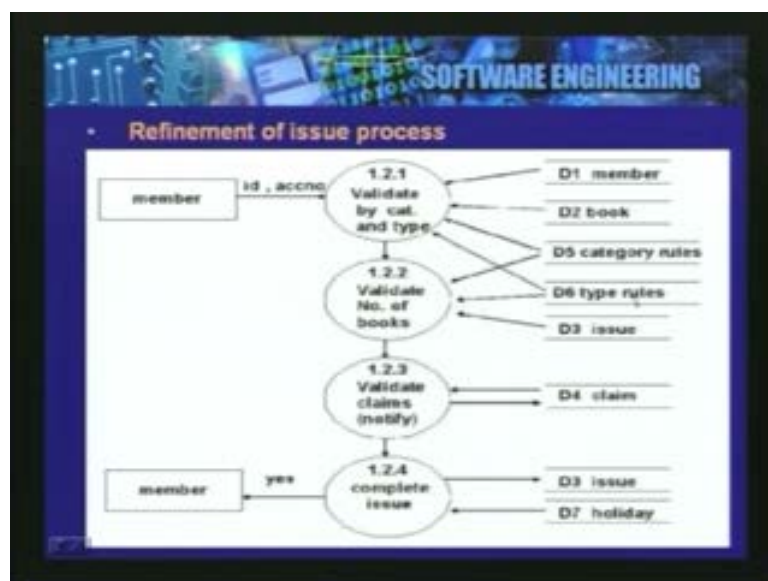


We have the issue data store which keeps record of all books issued, to whom the book is issued, the accession number of the book, the date on which it was issued and when the book is due; when the book is returned we also note down the date of return and if necessary we also will note here the fine which may be payable by the user.

We have the claim data store. The claim data store is basically supposed to keep the person who has put the claim, so member's id, assertion number of the book on which the claim is put and the date on which the claim has being put. So there may be many such claims for a given book and from a given user or a given member of the library.

Now that we have seen some of these data stores we can now refine the important processes in our data flow diagram so that the complex processes are specified in details now.

(Refer Slide Time: 05:55 min)



Now, issue process is one of the important processes; we will illustrate how the refinement for this has been done here. So we have the member who gives his identification and the books accession number that he wants to issue.

What happens in the process is that first we validate the category of the book and the type of the member whether this member is permitted to issue this book. Now to do this we need to bring data from the member, we need to bring data from the book, also data about the category rules and the type rule. Now these two data stores D5 and D6 are the new data stores that we have identified while decomposing. We need some place where these details of what category and what type of books can be issued so category rules will indicate how many books a particular category can borrow. The type will indicate what category of member can borrow that book and for how long he can keep that book. So these details are specified in terms of rules, there may be small tables which are stored as data stores here, they maybe actually fairly small data stores not containing much data but they are very important for executing this function.

So, after validating the category and type we validate how many books the person has already borrowed whether he is crossing the limit. So here we look at how many books are issued to him and again look at the rules. After that we validate with respect to claims; does the book have a claim of somebody else, if it has a no claim then we can proceed further but if it has a claim the claim maybe by the same member in that case again we can proceed further and in that case we may also have to adjust the claim data store; if the claim is by somebody else then probably the book cannot be issued to this student. So this way by decomposing we are introducing more and more details of the processing.

After this the book issue will be completed, we will note down the issue of the book, at this point we will also calculate the due date by which the book should be return and in order to do that we have to take into account the holidays so we identify one more data store called holidays. So, in decomposing the issue process we have completed the various details which need to be identified and which will be necessary later on when we develop software and implement the issue function. We may do similar refinements for other processes to clearly specify the details of actions that need to be carried out by the library in each case.

(Refer Slide Time: 09:04 min)

Slide 45 is titled "Requirements Analysis ..." and is part of a "SOFTWARE ENGINEERING" presentation. It features a blue background with a header image showing circuitry and the text "SOFTWARE ENGINEERING". The slide contains a bulleted list of notes:

- **Notes :**
 - claim data is modified if an issue is against claim made earlier
 - 'no' response could be given by any validate process (not shown)
 - D5 and D6 data stores define member categories and issue schemes
 - D7 contains holiday data for use in computing due date for return

The slide number "45" is visible in the bottom right corner.

Some important points are noted here; some of them we have already seen: the claim data needs to be modified when we are issuing a book in case there was a claim; in many of these we have not shown exceptions. The data flow diagram as we said earlier need not show every exception that is generally implied and we would be implementing them. So, for example, the type of book may not be appropriate for the type of the member. Now, in that case we will refuse the issue of book. Now that is not explicitly shown in the dataflow diagram. So we should show only those which are essential and others would be implied and would be implemented appropriately and we have identified some new data stores here which need to be created in the application.

(Refer Slide Time: 10:00 min)

Slide 46 is titled "Requirements Analysis ..." and is part of a "SOFTWARE ENGINEERING" presentation. It features a blue background with a header image showing circuitry and the text "SOFTWARE ENGINEERING". The slide contains a bulleted list of notes:

- **refine further where necessary**
- **interact with user to obtain procedural and data details**

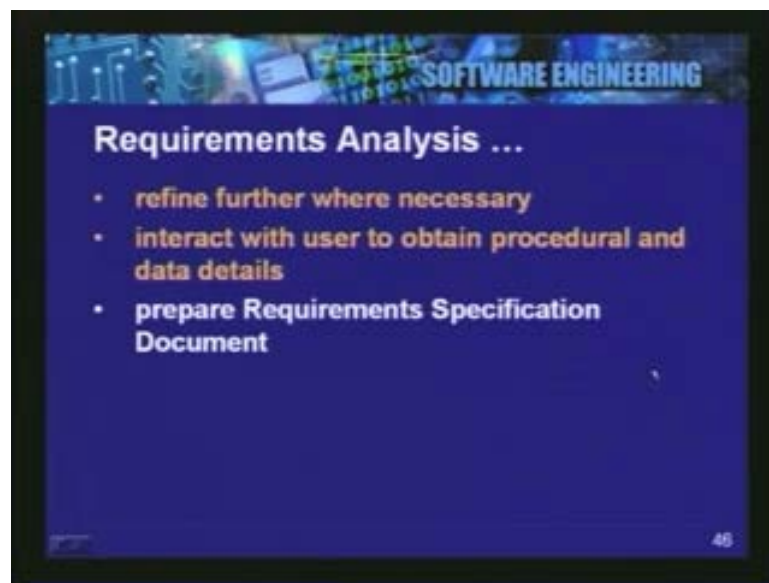
The slide number "46" is visible in the bottom right corner.

We will refine further if necessary. Now this is the decision that we need to take that do we have now sufficient details of each of these activities and if we are not refining let us say any of these subtasks further its procedural specification will also be recorded somewhere so that

we have a full level of details for subsequent design. This may be done for each vowel; for each vowel we may develop a short procedural description of what exactly is done, what data gets used, what data gets modified; this is the part of the algorithmic specification of that process.

Now that we have done all the analysis where we have found out the different activities, the different inputs, the different outputs and reports as well as the processing specifications we are now ready to write down the requirement specification document. This as we said earlier has a well-defined format. We had mentioned the IEEE format earlier; this is the base line, this has to be done thoroughly and properly.

(Refer Slide Time: 11:05)



Now, in this particular lecture we will give you an overview of this SRS document. Naturally we will not be able to spend too much time during the presentation itself; you should access this material subsequently. The course material will contain the details and you will need to study this in more so that you clearly understand what is specified in the SRS document to what extent it needs to be specified and how challenging it is to prepare a good SRS document.

(Refer Slide Time: 11:46)



So let us now see the document itself. We will show some important parts of this document. Here is the abstract; the abstract specifies the overview of the document itself. It says that the system to be developed will handle issuing of books to members of the library. The other important associated tasks are related to book claims and the fines, the document follows IEEE standards.

(Refer Slide Time: 12:22 min)



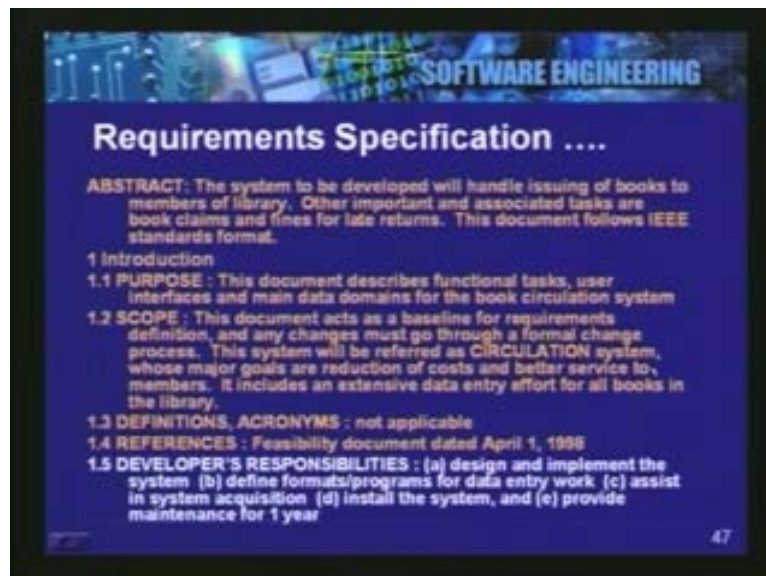
So now, in the introduction section we will first specify the purpose that this purpose gives the functional tasks that the users that the circulation system needs to implement, it also talks about user interfaces and we will describe the main data involved in this application. So this is the purpose of the SRS document that we have briefly stated here; we define the scope in section 1.2, this document acts as the baseline for requirement definition and any changes must go through a formal change process.

Now, with software, generally we have a change management or configuration management activity so we are indicating here that this is the baseline document **this is the** this forms a part of contract, any change must be done formally and should be accepted by all. The system is a circulation system whose main goals are briefly summarized here. It also mentions that the project includes an extensive data entry effort for all the books in the library.

If any definitions, acronyms need to be given they can be given here, we can refer to the feasibility document that we prepared earlier and what are the risks; the developers responsibilities?

This include design and implementation of system, defining the formats for data entry and also preparing the program for this purpose, assisting the librarian in buying the hardware, then installing the system and providing maintenance for the application for one year these are the clearly stated responsibilities.

(Refer Slide Time: 13:41)



SOFTWARE ENGINEERING

Requirements Specification

ABSTRACT: The system to be developed will handle issuing of books to members of library. Other important and associated tasks are book claims and fines for late returns. This document follows IEEE standards format.

1 Introduction

1.1 PURPOSE : This document describes functional tasks, user interfaces and main data domains for the book circulation system

1.2 SCOPE : This document acts as a baseline for requirements definition, and any changes must go through a formal change process. This system will be referred as **CIRCULATION** system, whose major goals are reduction of costs and better service to members. It includes an extensive data entry effort for all books in the library.

1.3 DEFINITIONS, ACRONYMS : not applicable

1.4 REFERENCES : Feasibility document dated April 1, 1998

1.5 DEVELOPER'S RESPONSIBILITIES : (a) design and implement the system (b) define formats/programs for data entry work (c) assist in system acquisition (d) install the system, and (e) provide maintenance for 1 year

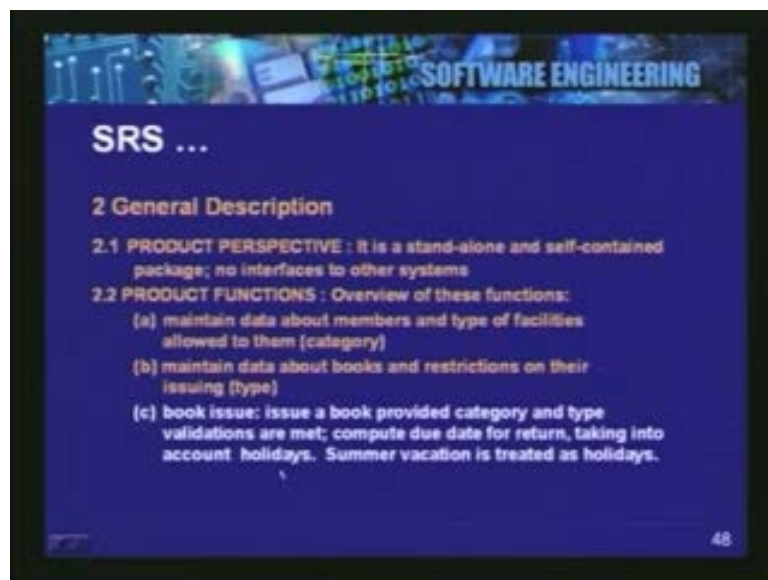
47

(Refer Slide Time: 14:01)



Then we give the general description about the product. The product is a standalone product there are no external interfaces. Then we summarize the functions of the product. Here I will illustrate only a few but you can proceed on the similar lines and define all the other functions which we identify. So here, since we are giving the overview we will just in one line try to state the important functions. These will be elaborated further in section 3 of the SRS which forms the body of the document.

(Refer Slide Time: 15:40 min)



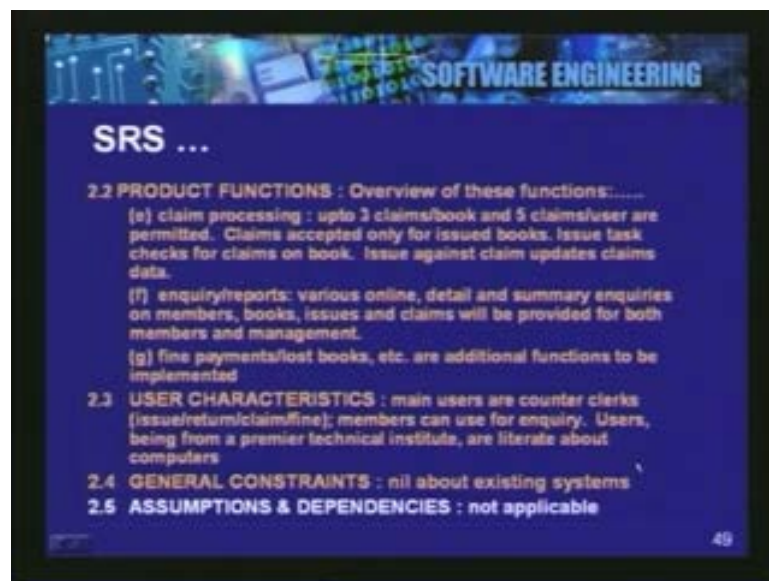
You should note here that it is possible to give all the summary functions, summary in the summary form in a short description. As you would see here, now we will list the functions of the applications we are building. the first is to maintain the data about the members of the library, these are the users of the library; to maintain the data about the books so new books will be added **books will be** if they become old or they are lost they will have to be removed so there is always some maintenance of master data that is involved. So these two functions

address the maintenance of member's data and the maintenance of the books data; then the book issue is another important function. Here we have noted that books are issued provided the category and the type are correct and we calculate the due date taking into account holidays. Summer vacation is also treated as a holiday. So, in a few words we are summarizing the essence of each function.

Then there is book return function at which time we will calculate fine if applicable, then we have a claim processing function; we are saying that up to three claims can be put per book and the user can put up to five claims, books can be claimed only if they are issued. If the book is currently in the racks then there is really no meaning in claiming the book. Then we will be supporting enquire and reports and finally we will be supporting the fine payment. And also if the books are lost their maybe some payment involved for that so all these details are broadly the main functions of the circulation application that we are developing.

So, in section 2 we have summarized them each of them in a few lines. We then note the user characteristics. We are saying that here the users are fairly computer literate and they will be trained but this will not be a challenging issue as such. Then any assumptions if they are applicable can be written down here.

(Refer Slide Time: 17:03)

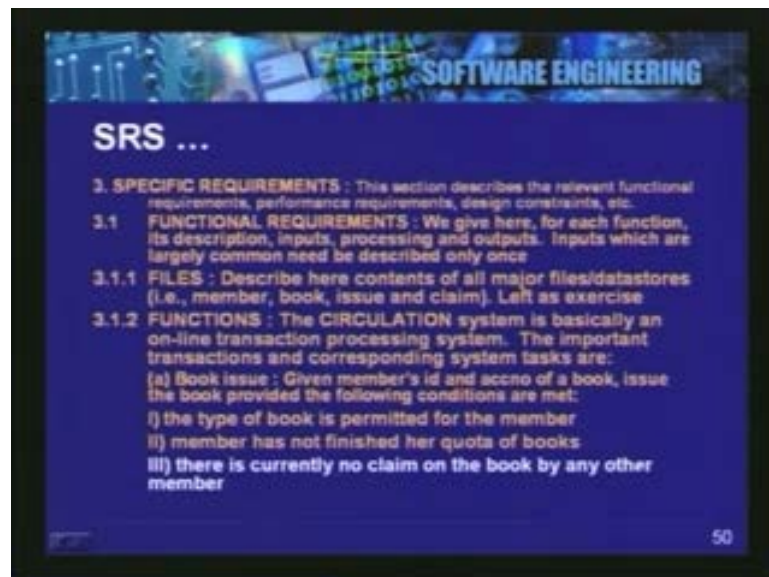


We then come to the important part the specification part. This part will describe each function in details to such an extent that we will be able to design and code each of these functions. So it will describe the inputs, the outputs and even the processing logic and any other design constraints all of these will be specified in section 3 so functional requirements should be complete.

We will first describe the various data stores that we have analyzed. Recall that for each data store we had identified all the attributes so they will be described here. Then each function will be described. The functions are issuing, returning and so on each of these will be described in full details. So here we are giving some details of the book issue function.

In the book issue function we are saying that we will be giving the member's id and accession number and then we will check the following conditions that the type of the book is permitted for the member, the member has not finished her quota of the books; there is currently no claim on the book by any other member. So if this is correct then the book can be issued and we will record it and the due date will also be calculated. The date due date will depend on the category of the member and the book type.

(Refer Slide Time: 18:22)



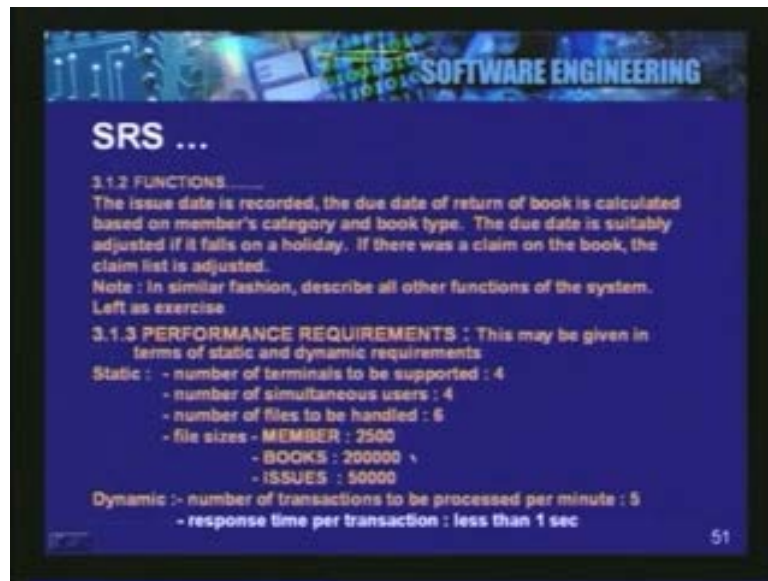
Holidays will also be taken into account and the claim list will be adjusted when the book is issued. So this way we describe all the other functions of the library and we will not go into the details of these; we will leave it as an exercise for you.

Next we also specify the performance requirements. Now, performance requirements can be given in two ways by indicating what are the static or the load requirements and what are the response time requirements for the different types of activities that we perform.

So we indicate here that the system will have three to four terminals, there will be four concurrent users, may be a few library clerks and few terminals will be given to users for making enquires directly by themselves. There are different types of files that we have already identified for the different data stores so we can then use their volumes.

For example, here we are indicating that we have about 2500 members, 2 lakh books and at any time generally 50000 books are issued. Now all these figures are important from the sizing point of view. What will be my requirements for the disk storage; what will be the requirements for my CPU and the main memory and so on based on which I can do the sizing format system based on the performance requirements. These are load requirements then we can indicate performance requirements. The performance requirements indicate here that we want the system to process about five transactions in a minute and that the response time generally should not be more than a second. Whenever we enter some data into the system and press the enter key to initiate that action we expect the machine to give its response in less than a second.

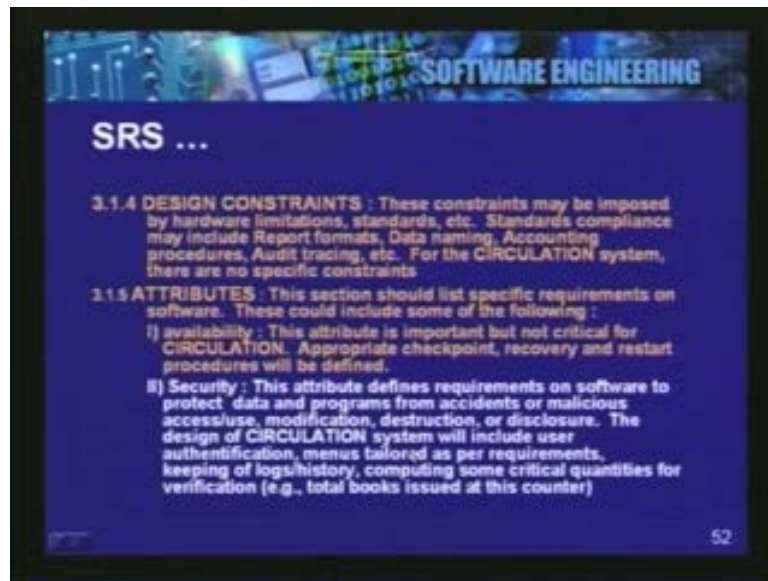
(Refer Slide Time: 20:52)



Now these requirements have been defined keeping in mind what the user expects from the system. Naturally they will have important implication on the sizing and the design of the system that we will make subsequently. The important point to note here is that these performance requirements are now an application for us and the user will accept the system provided we meet these performance requirements. So these also must be noted as part of SRS.

We may note any design constraints in terms of any existing hardware or standards for development and so on. In this system there are no specific constraints. Then we may include any other attributes which are from the availability point of view, should the system be available 24 by 7; naturally circulation system is not of that kind; although availability is important it is not so extremely important that we must provide a standby machine or some such thing. We are saying that availability will be addressed in the software; we will take an appropriate check points and we will provide facilities for recovering from failures and restarting of activities.

(Refer Slide Time: 22:12)



Security will also be important. Different types of users may access the system, there may be counter clerks, it may be the librarian or these maybe the members themselves who will be provided terminals. So we will provide appropriate access, restrictions or passwords and so on and the menu options that will provide to these different users will be appropriately restricted. So we note here the security concerns for the circulation system.

We may also indicate the maintainability requirements which indicate that we would develop the software keeping in mind the maintenance activities that may have to be done subsequently. So we will follow various well accepted norms and standards for developing a maintainable system. We also note down now the interface requirements.

Now, interface requirement the most important interface is that for the users of the library and this interface should be clearly specified. We had mentioned earlier that in a way this interface defines **in a in the user** some kind of a user-manual itself. Looking at this preliminary user manual the users will be able to get a good idea of the kind of software we are building for them. So these are external interface for the system for its various users. So we will have different interfaces to support for library users, separate interface for librarians, separate interface for the counter clerks who issue and receive books and separate interface for the users for enquiry purpose.

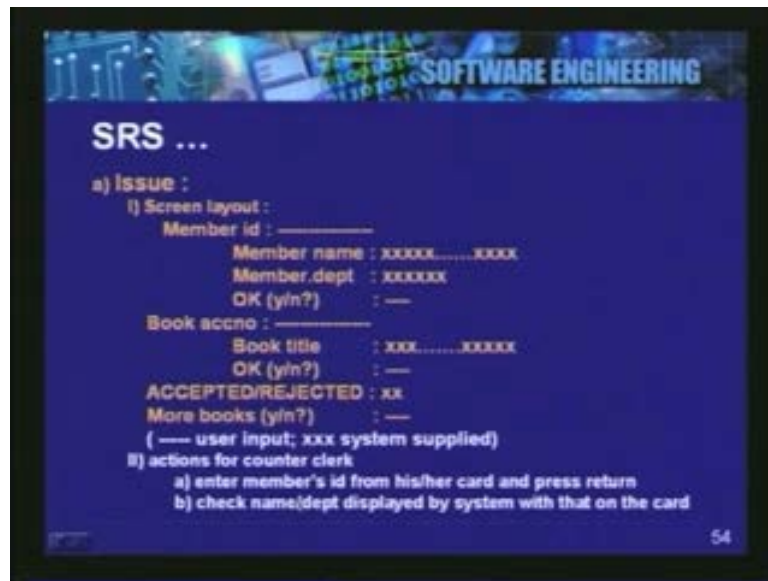
(Refer Slide Time: 24:03 min)



So this section is a mini-user manual by itself specifying the contents of various interactions. So we will describe this only for the issue function to give you a flavor of the kind of specifications we should make in this part of the SRS document. So, considering the issue activity we logically describe here the way the screen would look like, what will be the different elements that the screen will contain. So issue screen will accept the idea of the member for whom the book is being issued, then the system will respond by telling us the name of the user, his department and we will use this as verification.

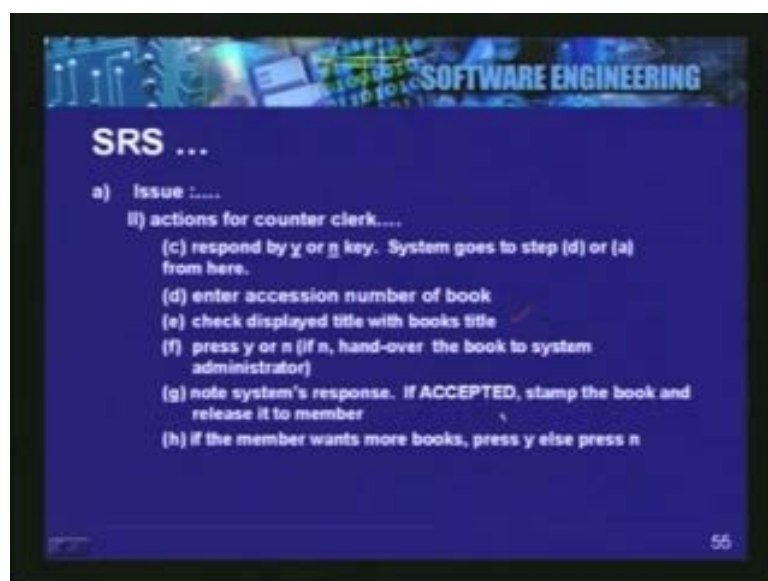
So the member id that we have entered is correct with respect to the system data then we will enter the book's accession number, the book's title will automatically appear on the screen we will verify that and then the system will indicate whether the request has been accepted or rejected. Rejection maybe for various reasons that we had identified earlier; if accepted actually more books may also be issued to the same user. So this is what..... these logical contents of the interaction would be for the issue transaction.

(Refer Slide Time: 25:36 min)



Now we also indicate the preliminary user manual for the counter clerk who would do this activity and this preliminary manual as you see here will list various things that the counter clerk has to do. It says that the counter clerk has to enter the member's id into the system, then check name displayed on the screen then if the system response is correct then the user will enter accession number, he will check the title of the book which is displayed on the system then if it is okay we will proceed further; then we will note down what is the system's response; if the system has accepted the issue of the book the user is supposed to put a stamp on the book. Now these are all external activities; now these external activities also need to be described in the preliminary user manual because this gives a clear idea to the library what is the system responsibility and what is the manual activity that still needs to be done in order to carry out this function.

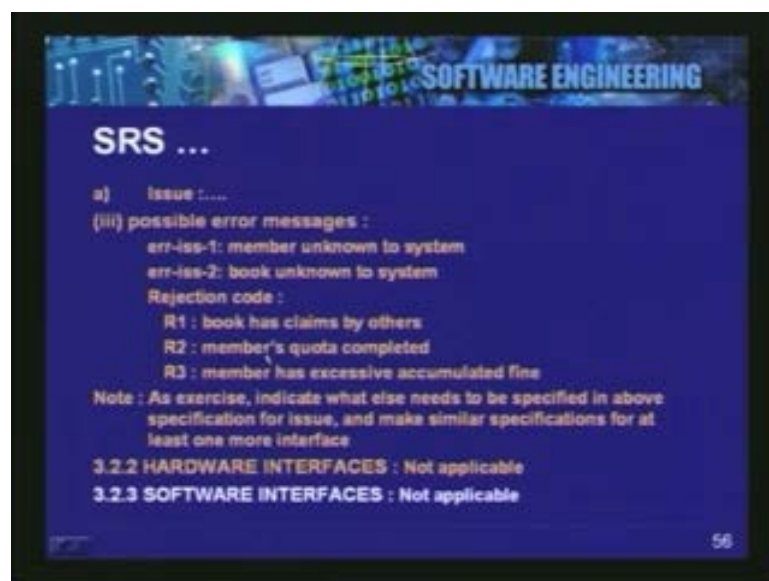
(Refer Slide Time: 26:41 min)



Therefore, the system says that the book can be issued. Now the book has to be physically stamped with the date and then it has to be handed over to the user. So this as you can see here is a very detailed specification of this particular function and it is described to an extent which gives a clear idea of how the system will function.

Then we also indicate the possible error messages which may take place when the issue activity is being done. There is an error which says member is unknown to the system. Error 2 says the book is also unknown to the system that means it is not entered into the library system. The transaction may be rejected and the rejections details are also given as a part of this interaction that the book has been claimed by somebody or member's quota has been completed or the member has excessive accumulated fine. These may be the different types of messages which may come from the system as a part of the issue function.

(Refer Slide Time: 28:05 min)



You should make similar specifications for other functions for the librarian. We then indicate hardware interfaces which are not applicable in this case; software interfaces are also not applicable because this is a standalone system. There are also there are no communication interfaces as such in this application.

We then note down any other requirements. In IEEE format some indications are given of what kind of requirements you should write down here. For example, these requirements could be about the data; what are the categories of let us say the retention, how long the data has to be written, what are the access capabilities and which functions access which data. So we will just indicate this specification for the illustration purpose.

(Refer Slide Time: 28:54 min)



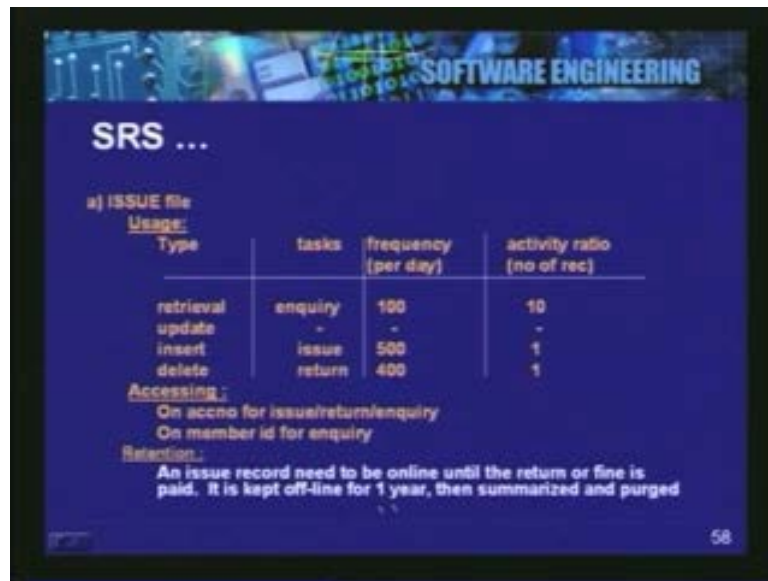
The various files have already been described. Now we will describe how these files get used in different functions and what is the amount of data that is accessed by them and what are the retention requirements; how long the issue data should be kept into the system. So, in this case let us see that we have the issue file which was described this issue file is used by retrieval type of or enquiry type of activities; how often do you have this activity happening in the library. We are saying that enquiries may be about hundred enquires happen and these enquires may read on an average ten records from the issue file.

Now this data will be very useful in order to estimate the performance requirements and what kind of system we may need to support this workload. So this is basically a workload specification; the issue file is also updated or it may have some new records being added so here we see that in issue file new records are added when we are undertaking the issue task and about five hundred books are issued generally in a day.

Similarly, deletes may happen say on an average maybe something like four hundred deletes happen. Of course there may be different frequency and different number of these operations on a given day but these are average figures.

We note down on which attribute the data is accessed from the issue file. So we note here that accession number is the most important field on which the data is accessed by all the other types of tasks whether is the issue task, return task or enquiry task and it may also be accessed on the member id; a person may want to know how many books have been issued to him and so on.

(Refer Slide Time: 31:14 min)



SOFTWARE ENGINEERING

SRS ...

a) ISSUE file

Usage:

Type	tasks	frequency (per day)	activity ratio (no of rec)
retrieval	enquiry	100	10
update	-	-	-
insert	issue	500	1
delete	return	400	1

Accessing :
On accno for issue/return/enquiry
On member id for enquiry

Retention :
An issue record need to be online until the return or fine is paid. It is kept off-line for 1 year, then summarized and purged

58

Then finally we note that the data in the issue file would be kept online till the book is returned or till the fine if any is paid. After the fine is paid or the book has been returned the **return the** issue records are kept offline for one year afterwards they will be summarized and removed from the system. So we are in fact noting down all different types of requirements of this application and these are recorded as per the format that IEEE has so carefully thought out. So **there has been**..... a place has been created for a type of specification in this format and we should ensure that all the requisite information for this application is captured and define in the SRS document.

We might also indicate what are the operations that users may have to perform as a special operation; is there something which is done at the start of the day or end of the day; do you verify at the end of the day the list of books which have been issued and which have been taken out from the library. So, these things maybe done as a one-time activity specifically so those are listed; some of these maybe done by the system some of them maybe done completely manually.

Similarly, we indicate how the user is expected to do regular backups and so on. We may also define some site adaptation requirements in case this system will be implemented in many libraries. Since we are implementing it only for one library this may not apply in our case. If it requires to be implemented at multiple sites we will have to give various adaptation requirements.

Next, we also identify the acceptance criteria. This is actually a very important criteria which will become the basis for acceptance by the user once the software is developed. So, at this point itself we are indicating what we will consider as our responsibility in helping the user in accepting the software. So we are saying that before acceptance of the system the developers will demonstrate all functionalities of the system for sample data of 100 books, 100 members and 2000 books. So we will indicate issues receiving of these books etc this will be demonstrated to the user.

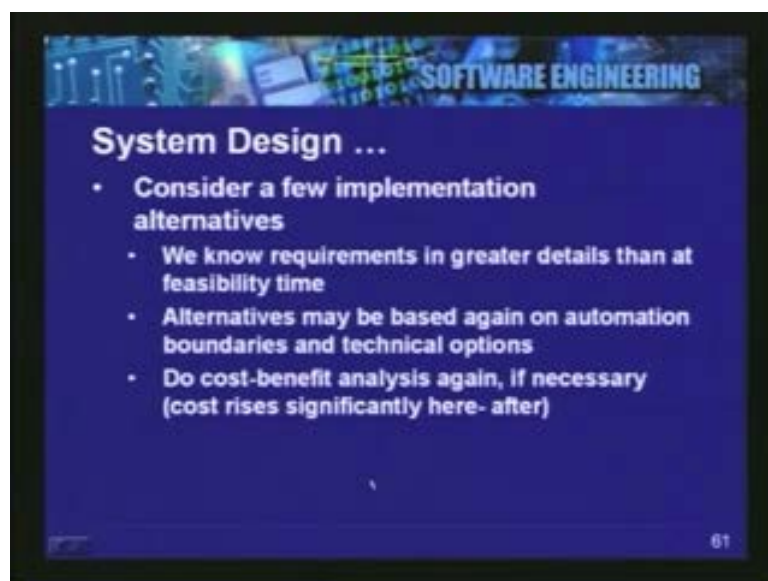
(Refer Slide Time: 33:57 min)



The developer will supply suitable test cases, expected results and this will be demonstrated to the users. Once **this will be** this is done we will assume that the acceptance has gone to satisfactory. This is then the SRS document. As you can see here all the analysis work that we have done which has helped us to get various kind of information has been now recorded formally in the SRS document. This SRS document will now be submitted and there will be a review the review would be done by the user they would tell us whether all the requirements have been captured properly it may also be reviewed by peers in terms of how good a job we have done in identifying various aspects of the system that we need to develop.

Let us assume that SRS has been accepted, it has been formally signed off then we proceed to the design phase. The high level design will consist of designing the software architecture, the files or the database and any associated manual procedures.

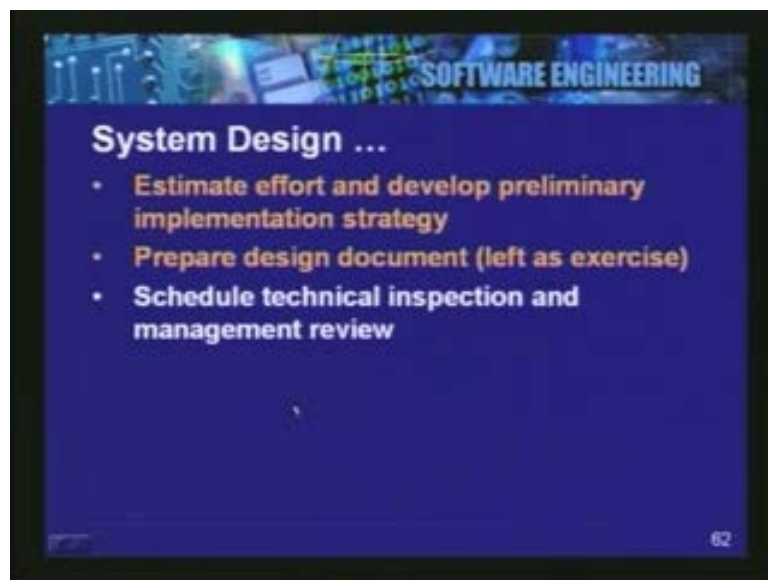
(Refer Slide Time: 35:10 min)



Since we are now going to put up lot of effort in design and implementation this is one more opportunity for us to **create** consider a few alternatives and if necessary do a cross benefit analysis. If we are considering alternatives we should probably do a full documentation of those and estimate the effort and prepare different implementation strategies which may be applicable for each of these alternatives.

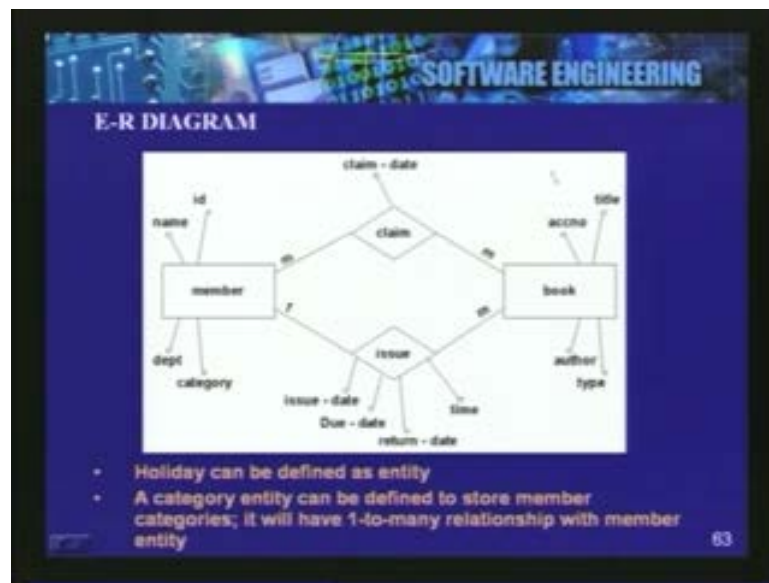
Once the design of the architecture, the database and the technical alternatives all these have been done we prepare the design document which we will not show in this particular case we leave it as an exercise but we had indicated what kind of format it has and this document will be reviewed. So we will now quickly take you through the different activities or different design goals that we try to meet during the design phase. Once the document is prepared we will have a technical inspection and management review as we reach the end of the design phase.

(Refer Slide Time: 36:20 min)



So we now are going into the technical phase where we will try to create the software architecture and the database design. For doing database design we will prepare the ER diagram.

(Refer Slide Time: 36:43 min)



The ER diagram is shown here. The two most important entities are the member and the book between them we have the issue relationship and the claim relationship. We can also create holiday as a separate entity; we can identify category of a member as entity, book type also has an entity and create one-to-many relationship between them. So this is only showing the important entities. You can complete this and show all the other entities and additional relationships.

Now this ER diagram then we will use for developing the database design. Very often you can do this through appropriate tools. Going from ER diagram to the normalized database design is usually a straightforward activity and we can look at the tables or the relations in the relational database model that we will derive from the ER diagram.

(Refer Slide Time: 37:34 min)



In fact a simple thumb rule could be that for every entity we have one table or one file and for every relationship also we have one file. So based on this we can prepare the table design; table design is what we do for relational database packages. So here are some of the tables that we will define.

(Refer Slide Time: 38:09 min)

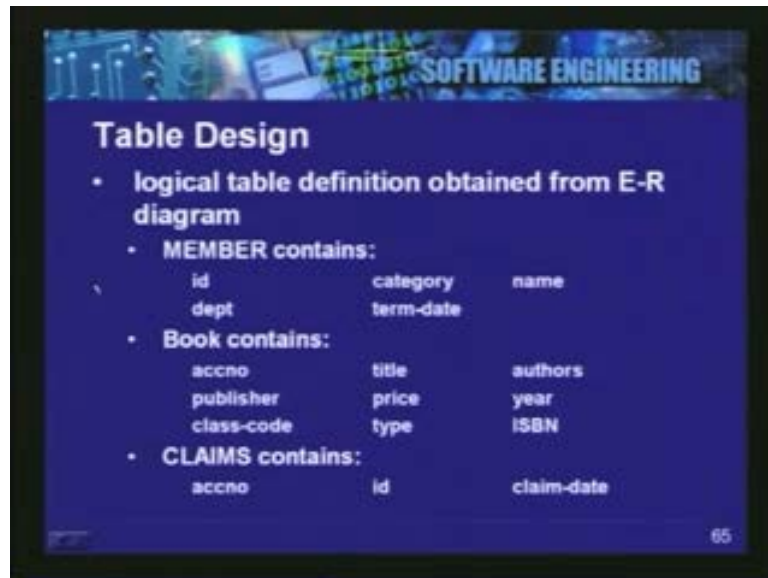


Table Design

- logical table definition obtained from E-R diagram
- MEMBER contains:**

id	category	name
dept	term-date	
- Book contains:**

accno	title	authors
publisher	price	year
class-code	type	ISBN
- CLAIMS contains:**

accno	id	claim-date
-------	----	------------

65

We have a table storing the member's data which comes from the member entity, tables storing the books data, the claims data the holiday's data and the category details. So these database tables can be identified as a normalized database design. Then we have the book type and issue. So these are the various tables which are required in these applications.

Now this is the normalized design. We will design the physical database from here which we will take into account the performance requirement, which we will take into account how the database is used so that we can ensure that the performance will be better with respect to where the data will be accessed.

Now this is generally called the denormalization kind of an activity where we look at the normalized database design and we made take some steps such as splitting a table or combining a few tables so that by doing this we reduce the number of IO operations or the disk operations that the application may have to do and in turn giving us a better performance. So we might look at splitting of a table so that the data which is accessed by the transaction is kept minimum or we may merge tables in case a transaction requires data from two or more tables. We may factor out data into a single record from multiple records or we may introduce additional fields; these are generally the summary fields. So these are various steps you take in de-normalizing database design and by doing this we try to make the different actions or different activities faster as it gives us a better response time.

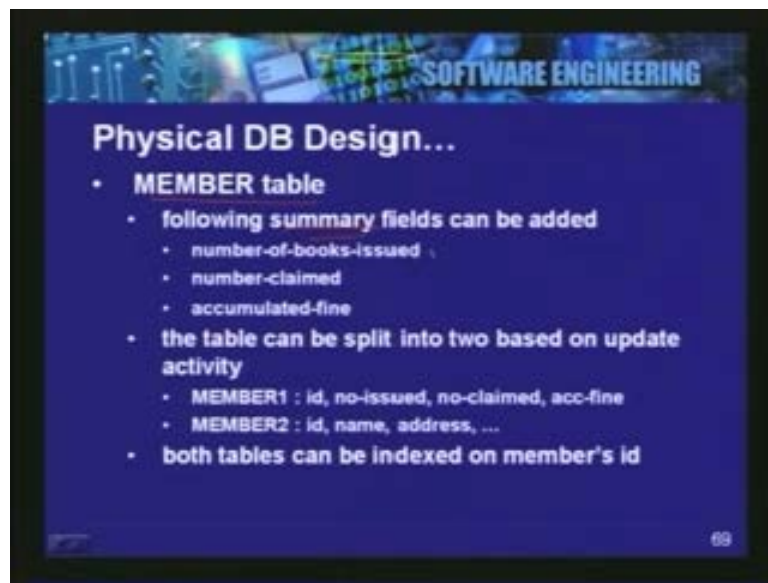
(Refer Slide Time: 40:31 min)



We will apply some of these in this case. After doing this we will also have to decide on what kind of file organizations and indexes we should create. Now we have sufficient data here to take these design decisions since we have already identified how the data is used by different activities such as issue, return and so on; we also identify on which attribute the data will be accessed. So we have all the information in our SRS document to help us in doing a physical database design.

The important point here to note is not exactly what we are doing in the circulation case study but that we can do all these activities because the data is captured in the SRS document and there is a set of techniques that we can apply for each of these design stages. We are doing logical database design now we are doing physical database design. For physical database design we have **certain acts** certain actions to be done based on the kind of data accesses which are made by different types of transactions. So there is a place for all these steps in the overall methodology.

(Refer Slide Time: 41:53 min)

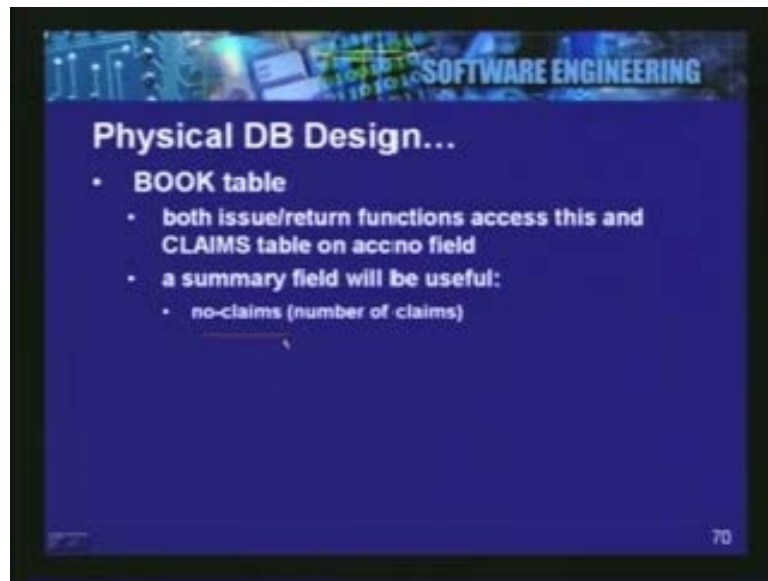


So let us look at the library application and what we are doing in the physical database design here. Now remember that we had a member table. Now, the member table contains data about members such as its member id, member name, and type and so on. Here now we note that some additional attributes can be added and these are summary type of attributes. Now these attributes include number of books issued to the member, number of books he has claimed, accumulated fine for this particular member.

Now, having this data in the member table itself would be very efficient because we will not have to count the books in issue file or we will have we will not have to count how many books he has claimed in the claim file; these can be directly obtained by accessing the member's data in the member file or the member table. So keeping these summary data for each member is going to make our issue transaction very efficient. So we introduce this summary data. But we must also remember that this summary data now needs to be updated every time we do issue or claim type of transaction so we may then decide to split the table into two tables called member one and member two.

Member one we will typically content the aggregate data or the summary data and the member two file will contain data which is not changing. So, both of these tables can be indexed on the member's id. This is how you do physical database design. You look at the table, you look at the different operations which are performed on this and in order to make those operations efficient you may introduce additional fields or you may split the file so that based on the different types of data access you may split that two different tables.

(Refer Slide Time: 43:57 min)



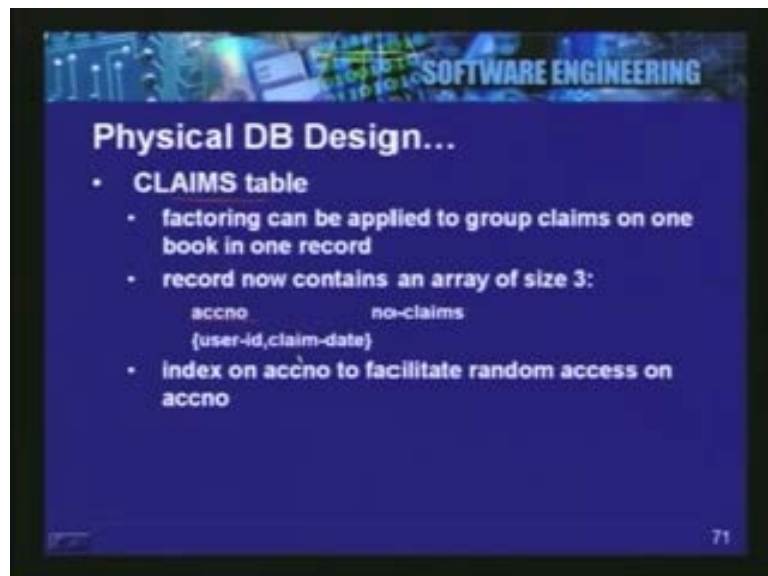
The slide is titled "Physical DB Design..." and is part of a presentation on "SOFTWARE ENGINEERING". It discusses the design of a "BOOK table".

- **BOOK table**
 - both issue/return functions access this and CLAIMS table on accno field
 - a summary field will be useful:
 - no-claims (number of claims)

The slide number 70 is visible in the bottom right corner.

Now we have given here design for other tables. For example, for the books file we will introduce a summary field called number of claims. In the claims table which had one record for every book claimed we may do factoring of data so that for a given book an array is kept which indicates all the claims for that book. So this particular record will have an array indicating the user's id and the date on which the claim was put for this book by different members. So this is going to be efficient, all the data will be available in a single record.

(Refer Slide Time: 44:21 min)



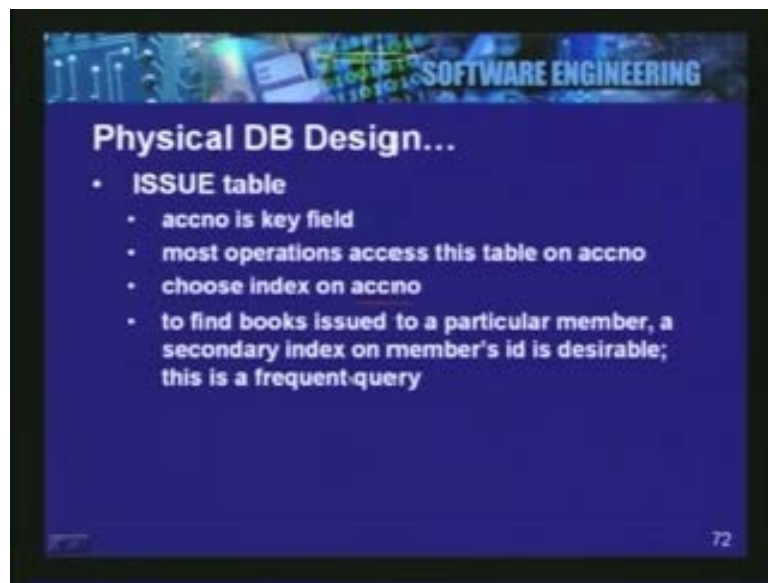
The slide is titled "Physical DB Design..." and is part of a presentation on "SOFTWARE ENGINEERING". It discusses the design of a "CLAIMS table".

- **CLAIMS table**
 - factoring can be applied to group claims on one book in one record
 - record now contains an array of size 3:

accno	no-claims
{user-id, claim-date}	
 - index on accno to facilitate random access on accno

The slide number 71 is visible in the bottom right corner.

(Refer Slide Time: 44:49 min)



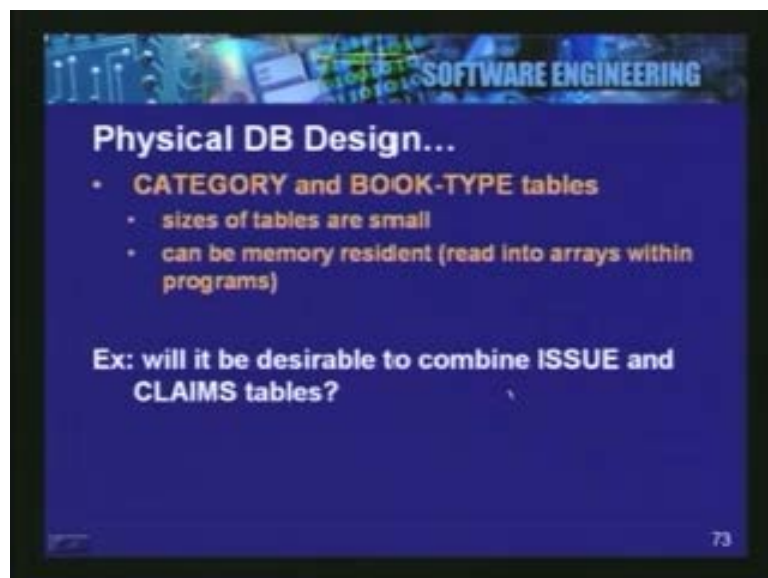
The slide is titled "Physical DB Design..." and is part of a presentation on "SOFTWARE ENGINEERING". It discusses the "ISSUE table" with the following points:

- **ISSUE table**
 - accno is key field
 - most operations access this table on accno
 - choose index on accno
 - to find books issued to a particular member, a secondary index on member's id is desirable; this is a frequent-query

The slide number "72" is in the bottom right corner.

Similarly, we apply design considerations on the issue table. We identify that this table has to be indexed on accession number and a secondary index may also be created on the member's id because this file has to be accessed frequently either on the accession number or on the member's id.

(Refer Slide Time: 45:19 min)



The slide is titled "Physical DB Design..." and is part of a presentation on "SOFTWARE ENGINEERING". It discusses "CATEGORY and BOOK-TYPE tables" with the following points:

- **CATEGORY and BOOK-TYPE tables**
 - sizes of tables are small
 - can be memory resident (read into arrays within programs)

An example question is posed: "Ex: will it be desirable to combine ISSUE and CLAIMS tables?". The slide number "73" is in the bottom right corner.

Then category and book type tables we note that they are very small and they can be kept in the memory itself. You can examine this question; can you combine issue and claims tables we lead be more efficient? So these efficiency considerations come in the physical database design. After this you do the software architecture design.

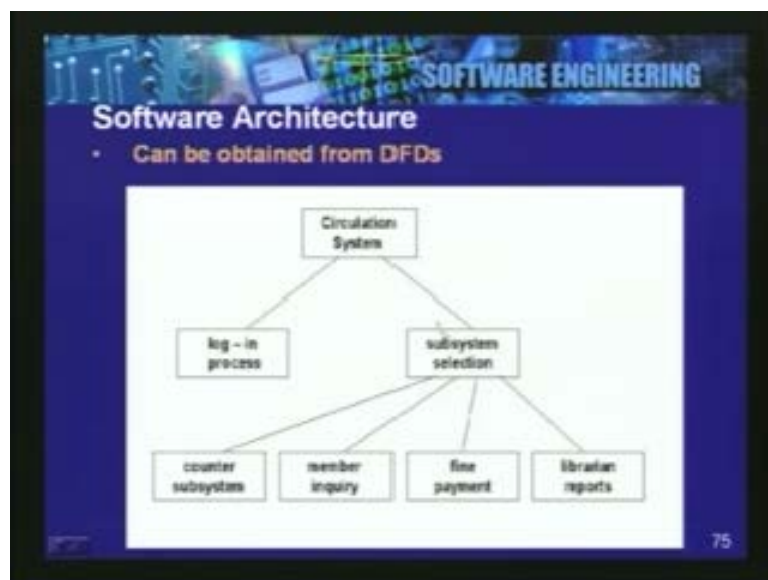
(Refer Slide Time: 45:33 min)



Remember that in the software architecture we try to identify the modules and see how they interact with each other by creating the software architecture diagram and you may obtain this from the dataflow diagram. We will show you some of the architectural diagrams for our library application. What you need to do is to study them in more details and also analyze them from that cohesion, coupling and complexity point of view. A module should not be very complex it should be cohesive and coupling should be minimized.

So we will show you some examples of software architecture for our application here.

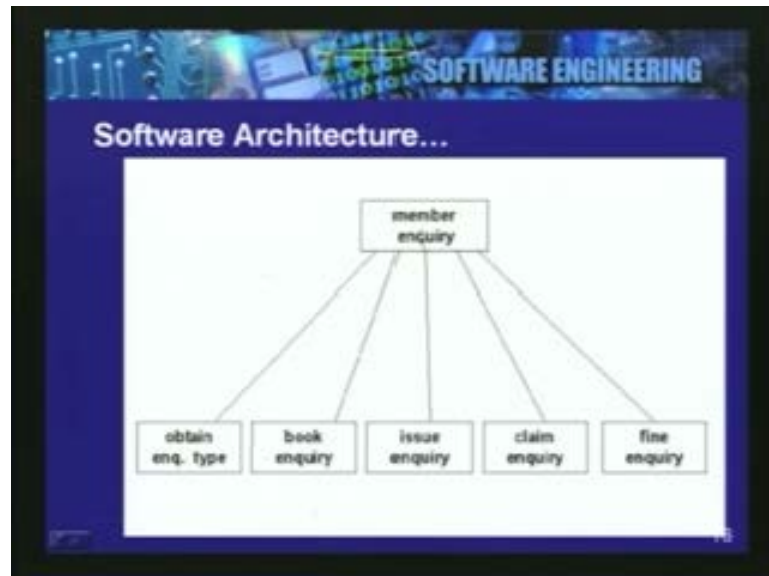
(Refer Slide Time: 46:35 min)



So here is the first level architecture which consists of a control module then it uses the login module and then there is a counter clerk type of a module which will basically select one of the modules based on the type of transaction we are processing. So it may be a member enquiry module, a fine payment module or it may be a request from the librarian or it may be

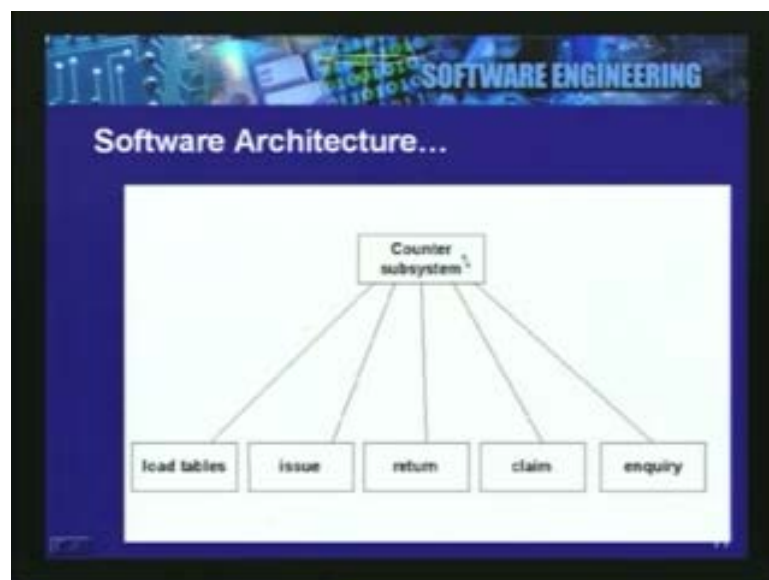
a counter clerk activity where we issue or receive books. So, at this point we have identified modules for different types of users. Then these will be further decomposed and designed into architecture for these subsystems.

(Refer Slide Time: 47:09 min)



So here is the member enquiry module which is further now decomposed and we find that this module can be written in terms of calling these lower level modules. Depending on the type of enquiry we may have a separate module for a book enquiry, issue enquiry, claim enquiry and so on.

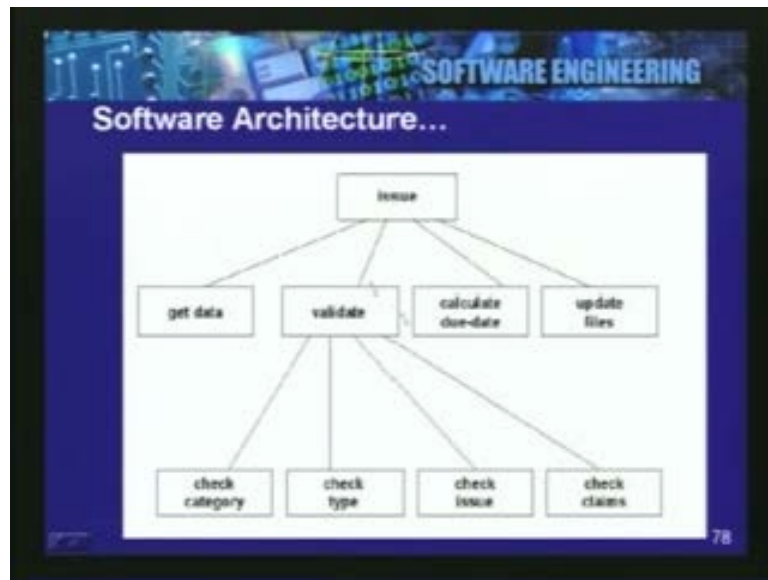
(Refer Slide Time: 47:35)



This is the counter subsystem module whose architecture now consists of different modules which load the various tables, which perform the issue transaction, return transaction and so on. So, each of them will be a module which will implement the functions required for these different transactions.

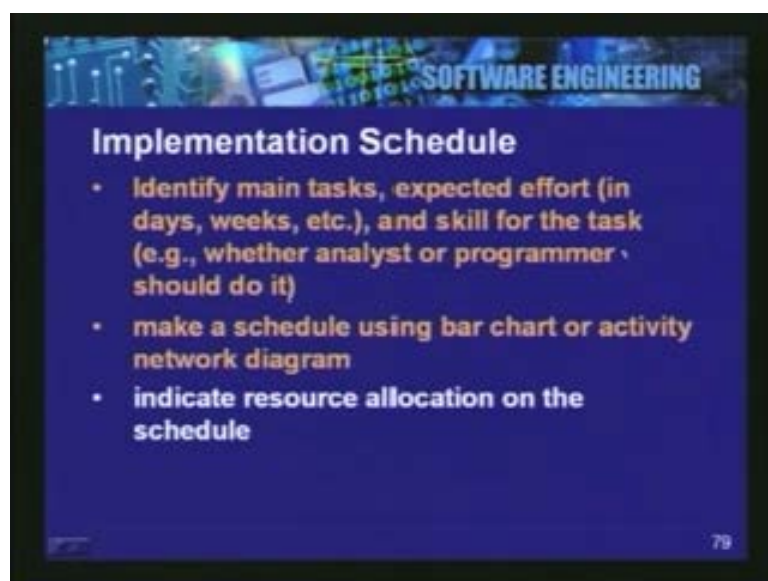
We had seen the detailed dataflow diagram for issue. So let us see the module architecture for the issue function. the issue makes use of these four modules at the next level getting the data, validating the data; validation consists of checking the category of the book, member type, then we check whether he has borrowed more books than.... Then has he already completed his quota and we check claims.

(Refer Slide Time: 48:28 min)



So, validate could do all these checks. Before going further and preparing for the issue of the book we will calculate the due date so we have a module for doing that and this module may actually have to be decomposed further in order to take into account holidays and finally we have the module which says update files. So here you see the software architecture in terms of different modules and how they could call each other.

(Refer Slide Time: 49:23 min)



After preparing the software architecture and the database design we can also prepare the implementation schedule where we identify the tasks, we estimate the effort and we also indicate the plan of how and who will perform these activities.

We prepare a schedule using a bar chart or a activity network diagram, we also indicate resource allocation on these so that we can clearly see the sequence of activities and how they will be completed. We also identify the activities which can be done in parallel. Once this is repair we can do a management review to ensure that the project will be completed in time. If we find that the time is not acceptable then we can actually add additional resources so that the implementation can be speeded up. So this is possible because we prepare an implementation schedule and analyzing.

(Refer Slide Time: 49:58 min)

Slide 80: Implementation Schedule...

- also indicate possible parallel activities
- during management review, we can decide on additional resources to speed-up implementation

80

(Refer Slide Time: 50:59 min)

Slide 81: Implementation Schedule...

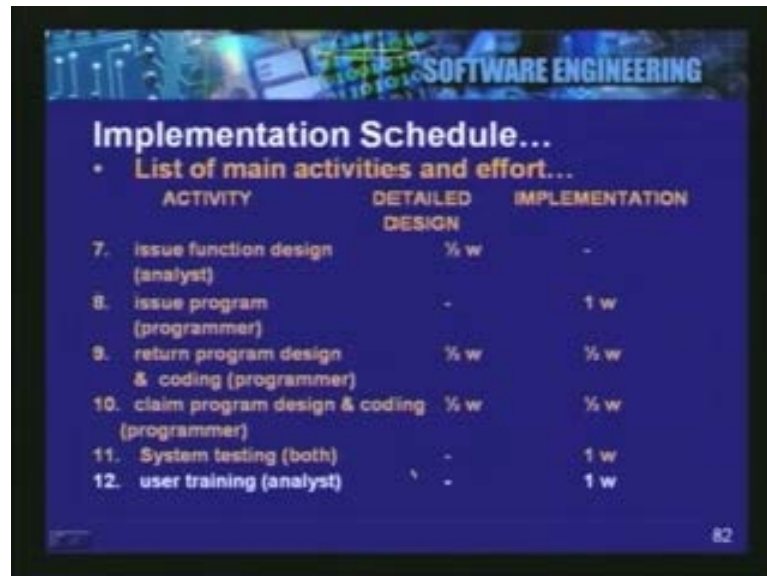
- List of main activities and effort

	ACTIVITY	DETAILED DESIGN	IMPLEMENTATION
1.	file definitions (analysis)	½ w	-
2.	data entry programs (programmer) for book data and member data	½ w	1 w
3.	management reports design (analyst)	½ w	-
4.	management report programs (programmer)	-	1 w
5.	member enquiry design (analyst)	½ w	-
6.	member enquiry programs (programmer)	-	1 w

81

So here is the list of various activities; you can go through these in details. But as you see, here we have listed the various activities that need to be performed starting from file design development of data entry program, then coding modules for management reports, now these reports will be first designed, these programs will be designed and then they will be coded. So, for each of them we have two types of activities: the further detailed design activity and then the coding and development activity. So all these have been listed and we have given time duration for detailed design as well as for implementation. So this is a detailed implementation schedule that we have prepared.

(Refer Slide Time: 51:20 min)



Implementation Schedule...

- List of main activities and effort...

	ACTIVITY	DETAILED DESIGN	IMPLEMENTATION
7.	issue function design (analyst)	½ w	-
8.	issue program (programmer)	-	1 w
9.	return program design & coding (programmer)	½ w	½ w
10.	claim program design & coding (programmer)	½ w	½ w
11.	System testing (both)	-	1 w
12.	user training (analyst)	-	1 w

Here are the other remaining activities. So we will quickly go through this. You should look at this material in more detail and understand it thoroughly in terms of its importance for timely completion of the project. So we will assume that we have one analyst and two programmers given that we can plan our implementation activities and assign the different activities to these people so that the work can be done in as much parallel as possible. We can estimate then the time to completion.

(Refer Slide Time: 51:41 min)



We then see how to prepare our design documentation. in the design documentation, as you recall, we will define the software architecture, we will define the database design, we will identify the different modules in terms of what functions they perform, what parameters they take, what output they produce so these could be a detailed design document that is prepared at the end of this and it is then reviewed technically. So this is an important milestone; after this we carry out the detailed design.

Detailed design we will prepare programmer specification and these detailed design document then will be given to the programmers to code the application. So design document is already ready from the previous phase. Now we specify each of the components in more details so inputs, outputs and processing were described. We now take each of the modules which is identified in the detailed design and develop programming specifications for each module in terms of data structure, internal logic and interface. Given this then the programmer will be able to implement.

(Refer Slide Time: 53:10 min)



The logic of the program can be given using a suitable pseudo programming language, some notation or a flow chart so that this can be handed over to the programmer. We will also do a detailed design walk through to ensure that the specifications are complete and consistent.

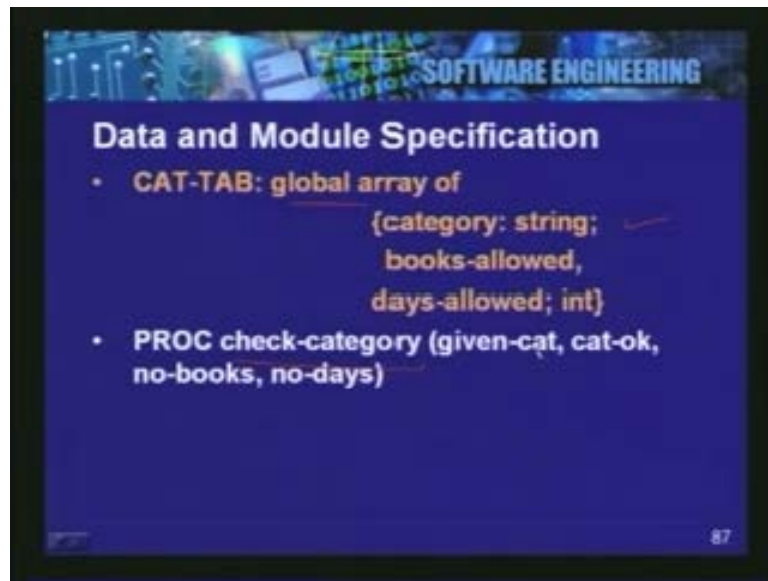
(Refer Slide Time: 53:37 min)



At this point we can evaluate module coupling and cohesiveness again to fine tune our first level design.

So we just illustrate here one specification of a module. This module indicates the important data structure which will store the category of the user and for the given category we will identify how many books are permitted to him and what is the duration for the issue.

(Refer Slide Time: 54:18 min)



SOFTWARE ENGINEERING

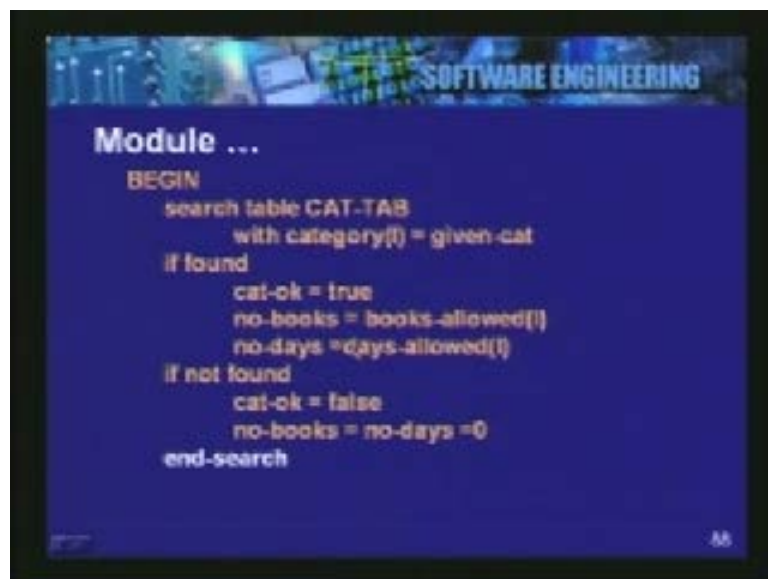
Data and Module Specification

- **CAT-TAB:** global array of
 {category: string;
 books-allowed,
 days-allowed; int}
- **PROC check-category** (given-cat, cat-ok,
 no-books, no-days)

87

Now this is a global array that the program will have and then we want to now specify a module called check category and this check category will take the inputs given here and could implement the logic of checking the array using the code which is described here.

(Refer Slide Time: 54:47 min)



SOFTWARE ENGINEERING

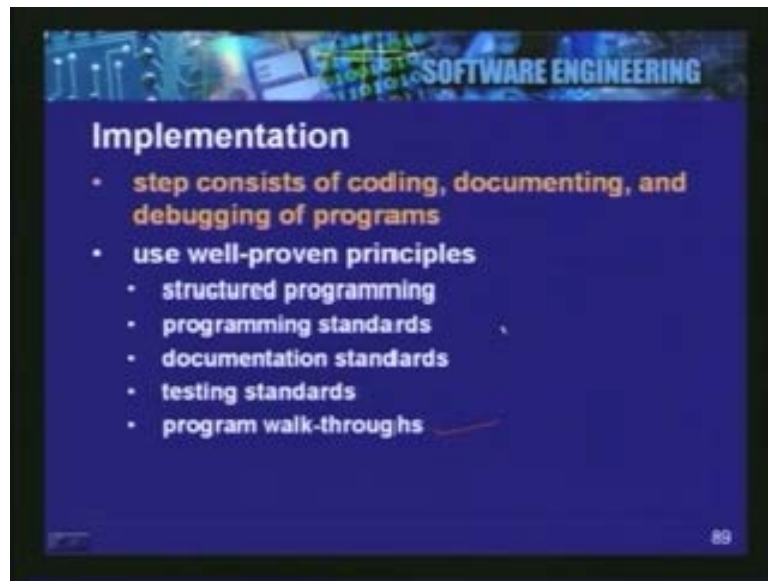
Module ...

```
BEGIN
  search table CAT-TAB
    with category[i] = given-cat
  if found
    cat-ok = true
    no-books = books-allowed[i]
    no-days = days-allowed[i]
  if not found
    cat-ok = false
    no-books = no-days = 0
  end-search
```

88

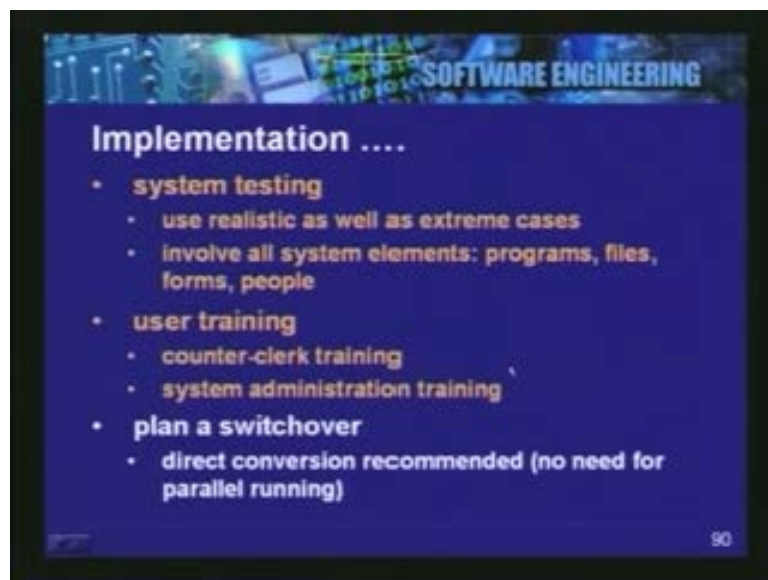
Now **this is the** this is the logic of this module; it says how the category table will be searched and how the information about how many books this member can issue and for how long we can issue will be described in a programming language form in this module. This detailed design now will be implemented. While implementing you will take into account the well-known principles of coding such as structured programming, you follow programming standards and you will document the code properly and you will also use the test cases which have been identified earlier and a program walk-throughs will also be done to ensure that your code is correct and it adheres to the organization standards so implementation will be carried out.

(Refer Slide Time: 55:28 min)



Now, these are highly technical activities and we expect that with your programming experience you will be able to carry these out properly. So we have gone through this implementation; now the testing phasing will be taken up; all the cases would be gone through, the test data has been prepared and test cases are also there; after this the system is completed, user training will be done, user manuals will be prepared and the acceptance test will be carried out; once the user has accepted a switchover will be planned.

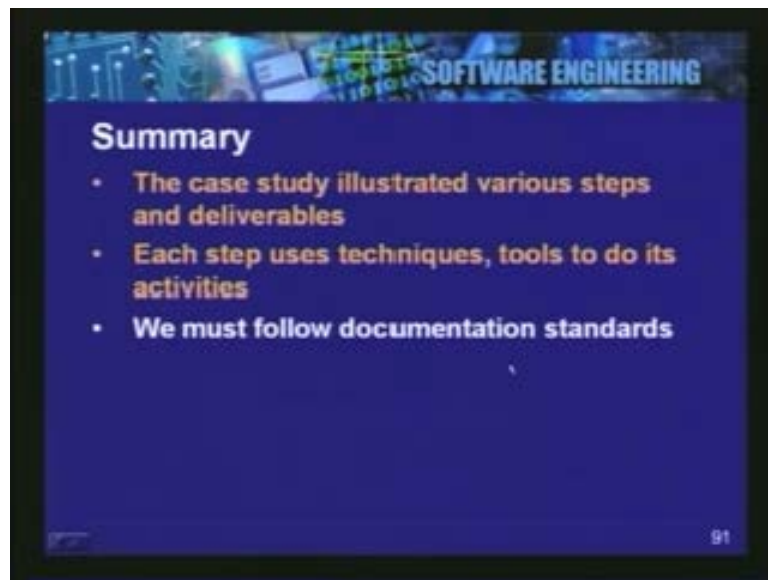
(Refer Slide Time: 56:05 min)



So these are the remaining activities of the methodologies through which we will go through. So we have covered this case study and we have looked at some of the important phases which are really the phases which identify the engineering aspect of the methodology that we must do the requirements very thoroughly, we must do the design systematically and these have to be properly documented and reviewed. So our case study has illustrated these different steps and different deliverables.

As we have seen the different steps used, various techniques and tools are very usable and they ensure that by applying those techniques we achieve the goals of engineering a good solution. So we have identified some of these techniques as we went through our case study. We noted the different documentation standards, it is very important that we follow them to ensure that all activities are done satisfactorily and documentation standards are defined for all the phases and the review ensures that these standards are followed.

(Refer Slide Time: 57:19 min)



Hence, review is a very important activity that ensures that the methodology has been correctly followed and all the deliverables have been properly defined. So this case study has illustrated the overall engineering approach to the software development and we hope that you will be able to do your development activities by following these systematic techniques.