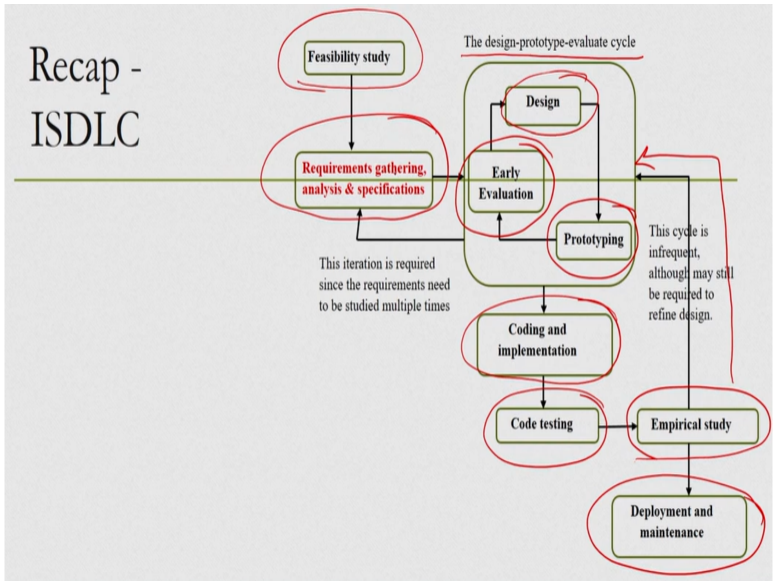
Design and Implementation of Human – Computer Interfaces

Prof. Dr. Samit Bhattacharya
Department of Computer Science and Engineering
Indian Institute of Technology, Guwahati

Module No # 02
Lecture No # 07
Functional specification of Requirements

Hello and welcome to the NPTEL MOOC's course on design and implementation of human computer interfaces. We are going to start lecture 7 which is related to requirement specification particularly specification of functional requirements.

(Refer Slide Time: 01:02)



Recap - ISDLC

Feasibility study
The design-prototype-evaluate cycle
Design
Requirements gathering, analysis & specifications
Early Evaluation
Prototyping
This iteration is required since the requirements need to be studied multiple times
This cycle is infrequent, although may still be required to refine design.
Coding and implementation
Code testing
Empirical study
Deployment and maintenance

If you can recall we are currently discussing the software development lifecycle stages where our primary concern is to develop interactive software. Now interactive software again just to recollect refers to a class of software that are supposed to be used by layman users who are not experts in the technology. For them we need to take into account various things most importantly user feedbacks at different stages of development.

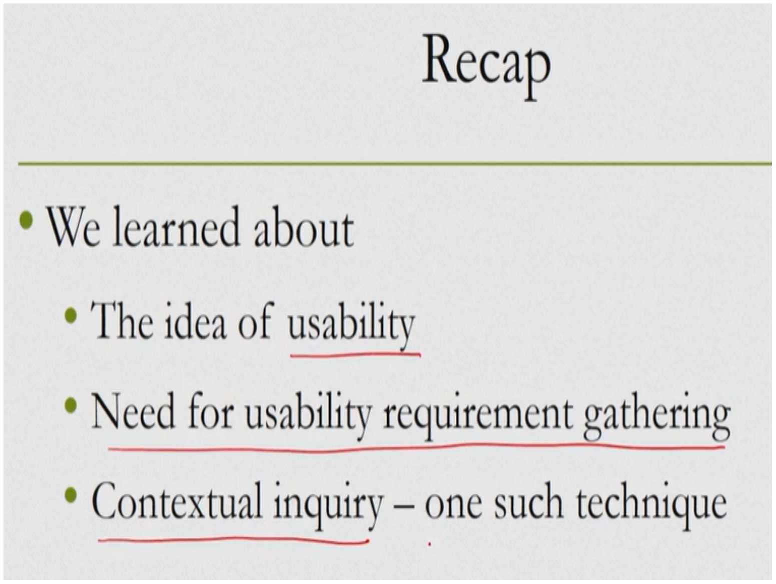
And in order to do that we have a special or a specific software development lifecycle meant for building interactive software and we are currently discussing different stages of the life cycle. To recollect there are several stages requirement gathering, analysis and specification is the stage from where we started our discussion. We have not discussed in details the feasibility study stage which is not very relevant for our core concepts.

After requirement there is this design stage, prototyping stage and evaluation of prototyping these 3 sub stages in a cycle constitute the design prototype evaluate cycle. Now here design refers to both design of the interfaces as well as interactions and also design of the system in terms of modules. Now those modules once finalized are to be implemented using programming some sort of programming language that is the coding and implementation stage.

Once the system is implemented we need to test it test the code that is the code testing stage after the entire system is coded implemented and tested. And we are to some extent assured of its executability we still need to test it for usability. Which is one of the key concerns in our interactive systems and that is done in the empirical study stage which is expected to be done once or twice at the most because this stage is quite costly.

However there is a possibility that output of this stage can lead here and there can be a cycle. So once this study stage is complete and we are assumed of the usability as well as executability of the system we go for deployment and maintenance that is our last stage. Currently we are discussing the requirement, gathering, analysis, and specification stage.

(Refer Slide Time: 04:11)



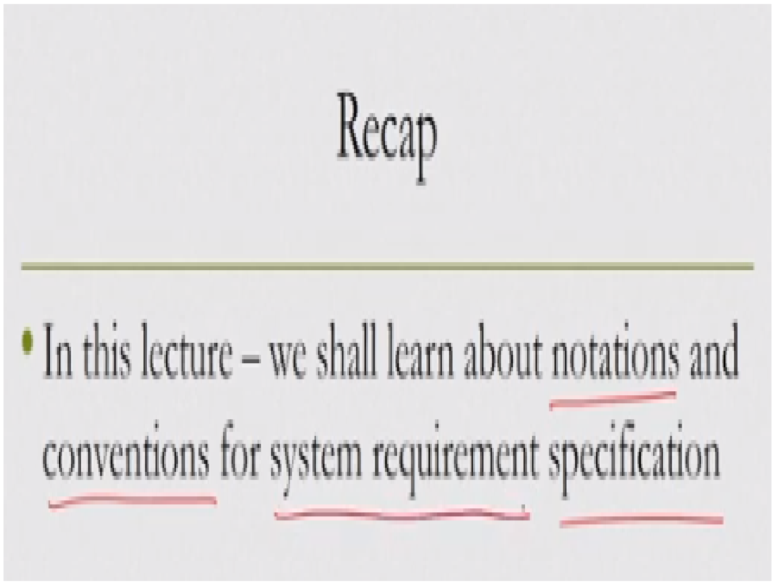
Recap

- We learned about
  - The idea of usability
  - Need for usability requirement gathering
  - Contextual inquiry – one such technique

Now we learned about usability earlier which is one of the crucial aspects of the design of interactive system. We also have learned about the usability requirement gathering why that is important and how it helps building a system that is likely to be acceptable to the end users? We have gone through in the previous lecture one method of requirement gathering particularly usability requirement gathering that is contextual inquiry.

Again it may be noted that this is only one of many such techniques however we discussed this technique contextual inquiry technique in some details.

(Refer Slide Time: 05:01)



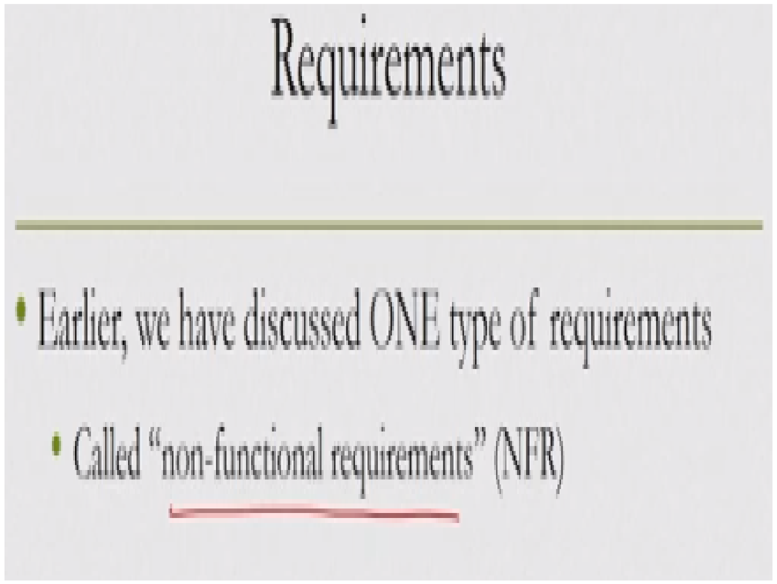
Recap

- In this lecture – we shall learn about notations and conventions for system requirement specification

Now contextual inquiry technique; is used to gather the requirements. And we have also seen how these requirements can be analyzed whatever we have gathered during contextual inquiry can be analyzed to identify design goals or some such things through the reflect stage using various tools such as affinity diagram method. So we discussed in details affinity diagram methods with examples to understand how the observations that are made during contextual inquiry can be converted to some design guidelines or something that can help us design usable product.

In this lecture our focus is actually how to specify those designs so that other team members can make use of the specification in subsequent stages of the life cycle. So this lecture will primarily focus on notations and conventions that are used for specification of system requirements that is the subject matter of this lecture.

(Refer Slide Time: 06:20)

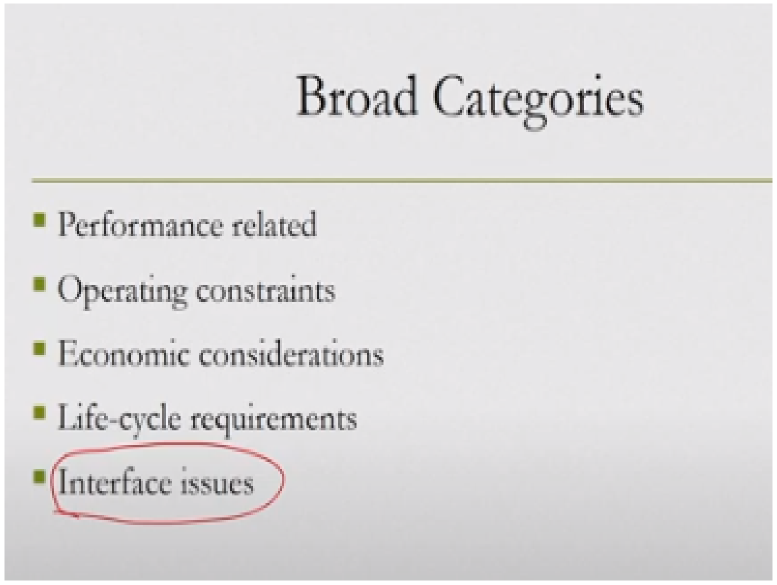


Requirements

- Earlier, we have discussed ONE type of requirements
  - Called "non-functional requirements" (NFR)

Now before we go to the actual subject matter let us quickly recollect that whatever we have termed as usability requirement earlier is also known as non-functional requirements. This is a term that is used to refer to requirements such as usability and some other things so usability requirements are one type of non-functional requirements. So what are the other non-functional requirements let us have a quick recollection of those types.

(Refer Slide Time: 06:58)

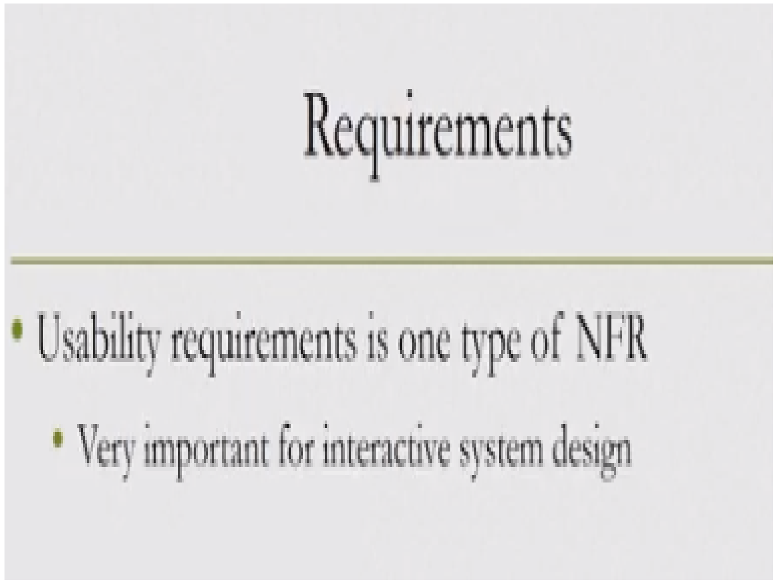


Broad Categories

- Performance related
- Operating constraints
- Economic considerations
- Life-cycle requirements
- Interface issues

So there we can have performance related non-functional requirements operating constraints that can be listed as non-functional requirements. Economic considerations which are again considered to be non-functional requirements life cycle requirements and finally interface issues that are non-functional requirements. Now among these interface issues both hardware and software interface issues are used are referred to and the software interface issues are generally termed as usability which belongs to this kind of non-functional requirements.

(Refer Slide Time: 07:44)

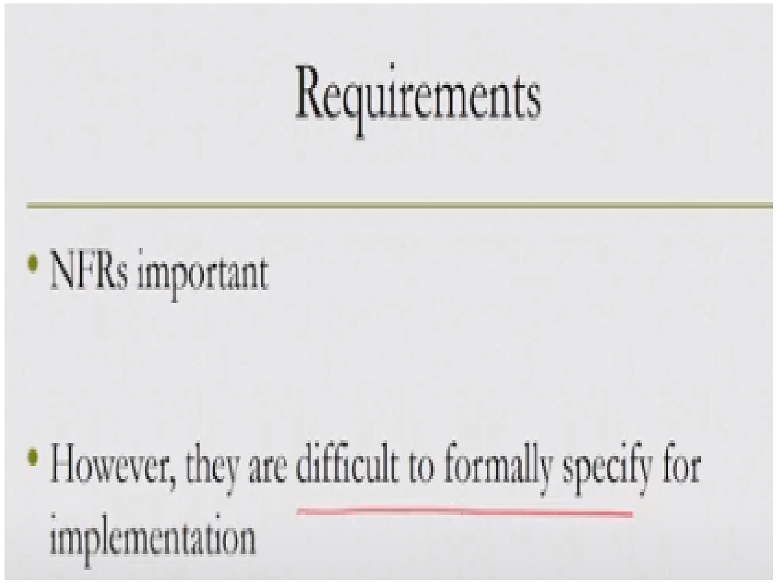


Requirements

- Usability requirements is one type of NFR
  - Very important for interactive system design

And we have repeatedly emphasized that usability is one very important non-functional requirement that has to be taken into account while we are going to build an interactive system. So we have discussed several non-functional requirements which are very important in the context of system design.

(Refer Slide Time: 08:11)



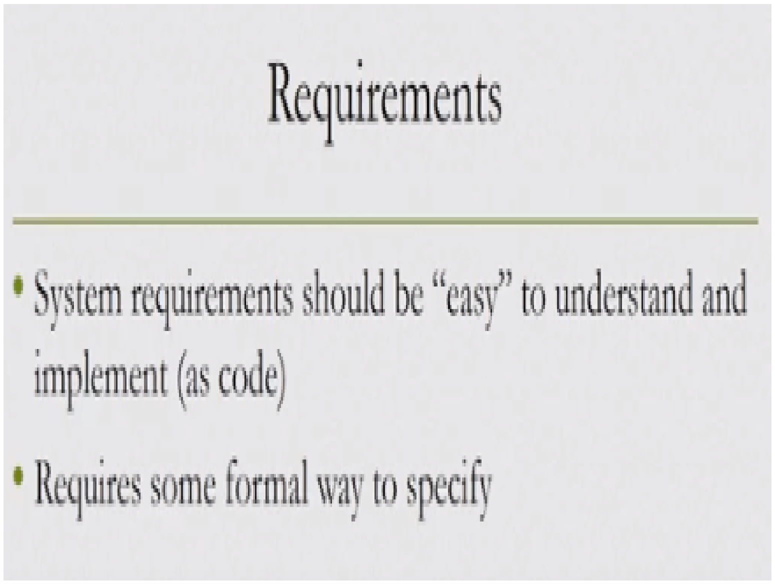
Requirements

- NFRs important
- However, they are difficult to formally specify for implementation

And in particular we have discussed usability which is very important in the context of interactive system design now they are of course important. But one issue with non-functional requirements only is that these are difficult to formally specify and unless they are properly specified. Unless we are able to specify the requirements in a proper way a suitable way it is difficult to convert them to design requirements or into a proper implementation of the system.

So we need to somehow be able to specify these requirements in a proper way but unfortunately non-functional requirements are not very much suitable to be converted to that proper way. So that we can immediately be able to convert it to some implementation or some design of the system.

(Refer Slide Time: 09:12)

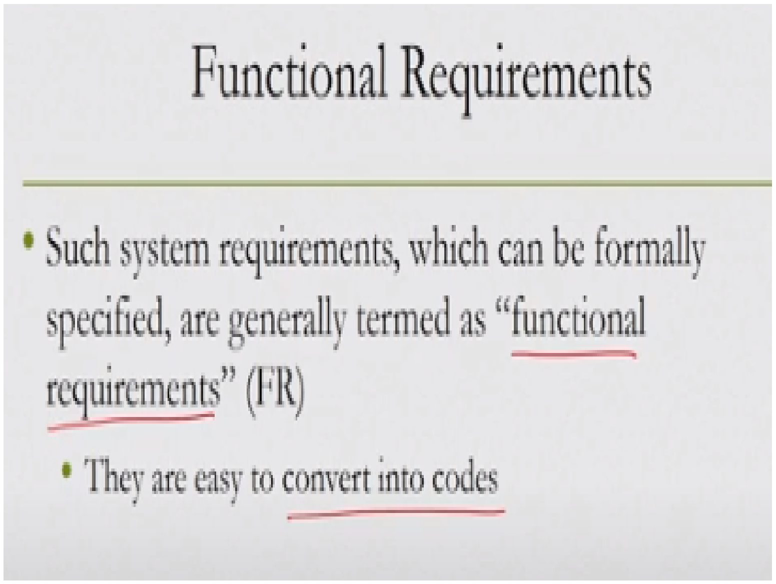


Requirements

- System requirements should be "easy" to understand and implement (as code)
- Requires some formal way to specify

In fact whenever we are talking of system requirements and this next stage is implementation then our objective should be to have requirements which are easy to understand and implement in the form of a program. That is one sort of requirement of the requirements so to speak that we should aim at. And in order to achieve that what we need is some formal way to specify the requirements that is required.

(Refer Slide Time: 09:50)



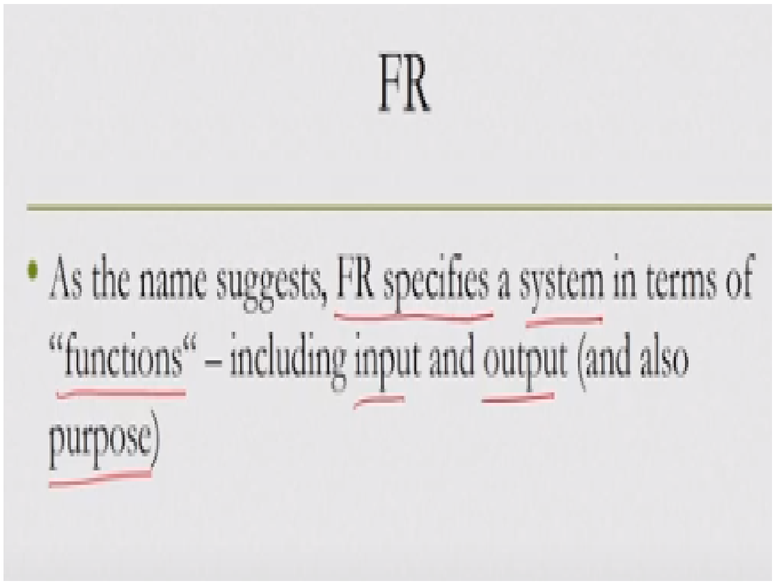
Functional Requirements

- Such system requirements, which can be formally specified, are generally termed as "functional requirements" (FR)
  - They are easy to convert into codes

Now when we say that we want to formally specify the requirements that means we want to make it easy for the team to implement the next stage of the development lifecycle by team I am in the development team. And when we are able to do that when we are able to specify requirements in a somewhat formal manner for quick implementation those requirements are generally called functional requirements.

They have these properties that they are easy to understand and convert to codes we shall see with some examples what we mean by these terms that convert to codes easy to understand and so on.

(Refer Slide Time: 10:41)



FR

- As the name suggests, FR specifies a system in terms of "functions" – including input and output (and also purpose)

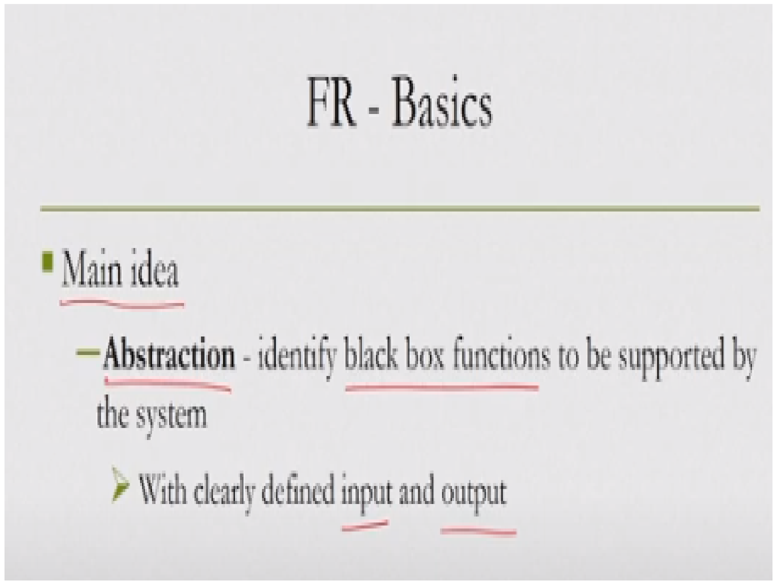
Now as the name suggests so the name is functional requirement so as the name suggests functional requirements specify a system or the requirements of the system in terms of functions. What are functions? Mathematically functions are defined in terms of its input output and here also we specify the requirements in terms of functions that take some specific input produces some specific output and these functions also typically has some purpose mentioned in the specification. So in functional requirement we have functions defined in terms of input output and purpose.

(Refer Slide Time: 11:35)

Functional Requirement Specification

So let us depart into the idea of functional requirement specification what are those and how to specify?

(Refer Slide Time: 11:50)



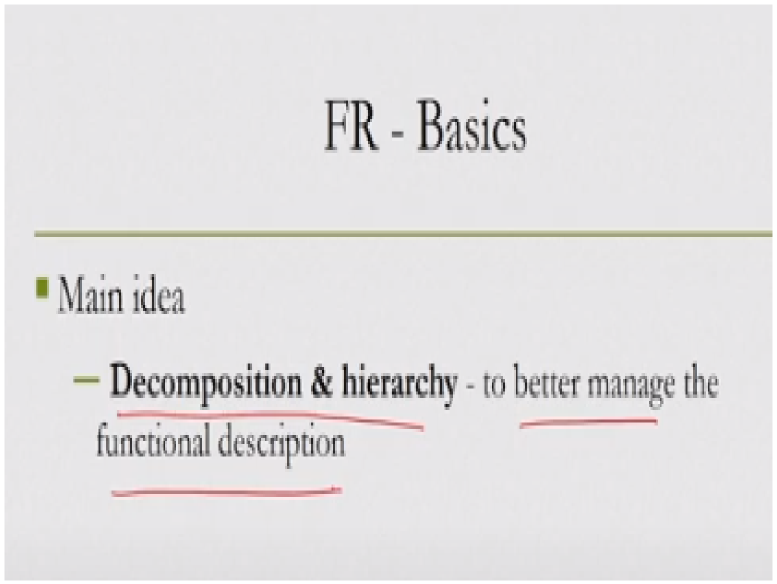
FR - Basics

- Main idea
  - **Abstraction** - identify black box functions to be supported by the system
    - With clearly defined input and output

Now before we are able to create a functional requirement specification we should be aware of few things first of all when we are talking of functional requirement specification the main idea is abstraction. What is meant by this term is that? We only need to identify black box functions to be supported by the end product or the system. And in this black box functions we need to only specify the input and output.

So, in other words abstraction tells us that identify only the need rather than how to achieve it. So we do not know how the function can be implemented or defined rather only what input it will take and what output it should produce. So the function in itself appears to be a black box with only an input and output specified rather than how the function should be implemented? So that is the first thing that we should keep in mind here we are trying to only identify functions as black boxes rather than trying to design algorithms to implement the functions.

(Refer Slide Time: 13:07)



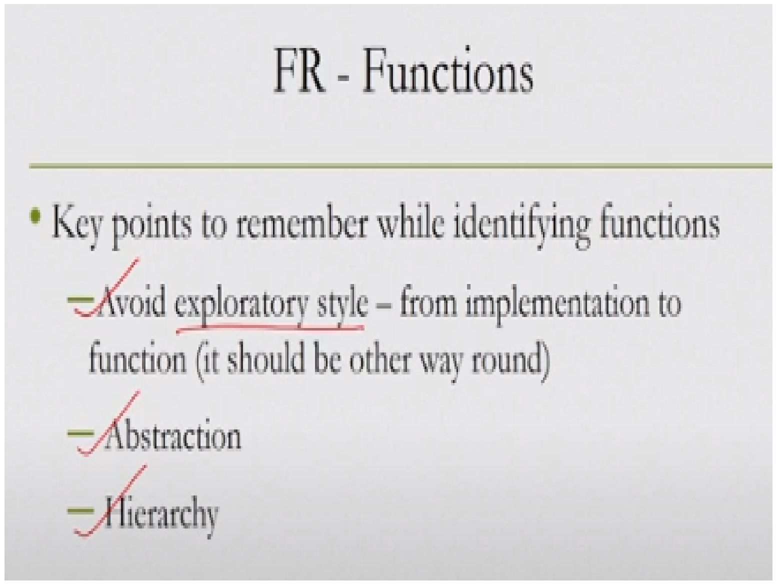
FR - Basics

- Main idea
  - **Decomposition & hierarchy** - to better manage the functional description

Second thing is decomposition and hierarchy so when we are trying to identify the functions we should decompose the entire requirement into some sort of hierarchy of functions and sub functions. Otherwise it will be difficult and too complex to specify the functions. If we can think of hierarchical representation of functions and sub functions then it will be easier for us to specify the function input and output much more easily than if we do not follow any hierarchy.

So essentially this decomposition into hierarchical levels and the levels of the hierarchy help us better manage the description or the functional description of the system. So we need to have abstract notion of function as black box with only input and output. And also we need to think of the functions in the form of a hierarchy so that we are able to better manage it easily define it and so on.

(Refer Slide Time: 14:22)



FR - Functions

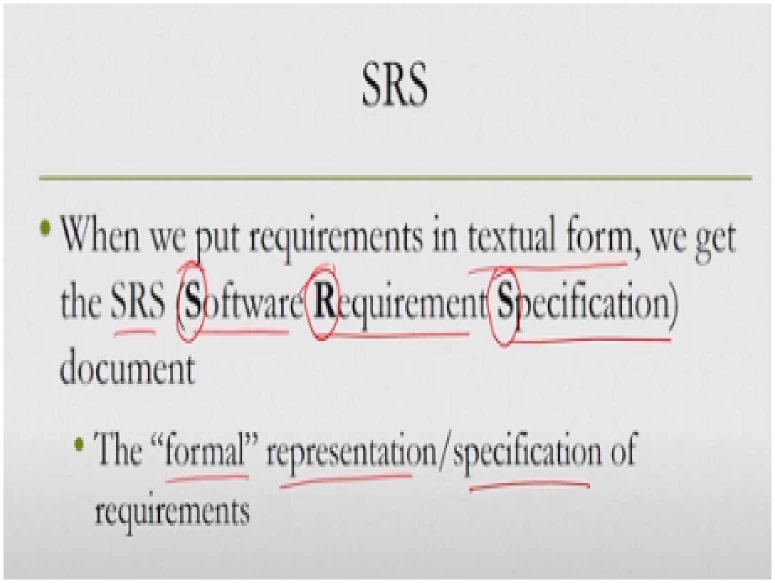
- Key points to remember while identifying functions
  - Avoid exploratory style – from implementation to function (it should be other way round)
  - Abstraction
  - Hierarchy

So what are the key points that we should remember while defining these functions? First thing is we should avoid exploratory style of definition what it means? It should not be the case that first we should think of implementing the functions and then from there we go back and think of creating the functional hierarchy. That is exploratory style first we keep on implementing and then from that implementation we learn about the presence of the functions.

It should be the other way around first we should think abstractly the functions and then think of how to implement them in a letter design stage. Abstraction as we have already pointed out the function should be thought of in a very abstract form. We should not start thinking about the algorithm to implement the functions rather we should think of functions as black boxes and only input and output need to be thought of plus the purpose of the function.

And finally it will be easier to manage the specification if we can think of the functions in the form of a hierarchy of functions and sub functions rather than only one level that will make things easier to manage. So these 3 things we should keep in mind we should try to avoid exploratory style, we should think of functions as black boxes and we should try to think of a functional hierarchy to represent the all the functions in the system.

(Refer Slide Time: 16:04)



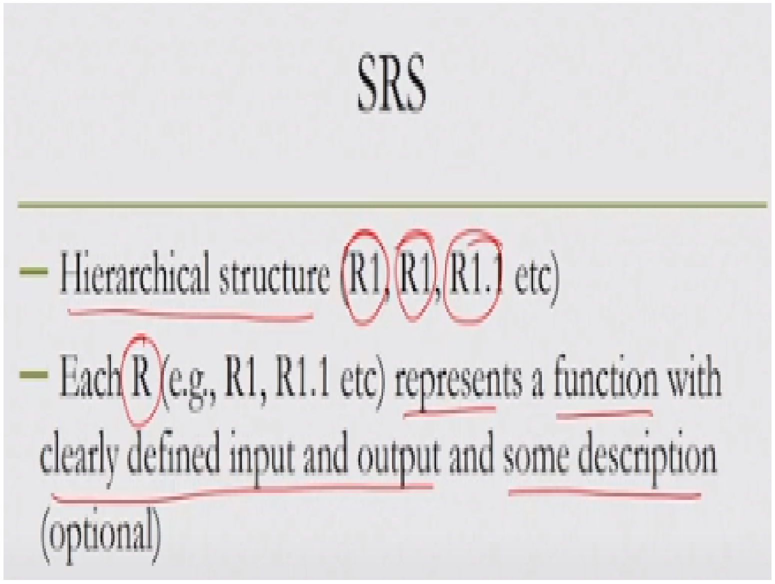
SRS

- When we put requirements in textual form, we get the SRS (Software Requirement Specification) document
  - The "formal" representation/specification of requirements

Keeping these in mind we should come up with a functional hierarchy that should be our end product in the functional requirement specification stage. And when we put these requirements in textual form in some specified format what we get is called the SRS or software requirement specification document. So SRS this acronym comes from these terms these words software requirement specification.

Now this SRS document is essentially our quote unquote formal representation or specification of the requirements. So our objective is to create an SRS document after this functional requirement specification stage is over.

(Refer Slide Time: 16:59)



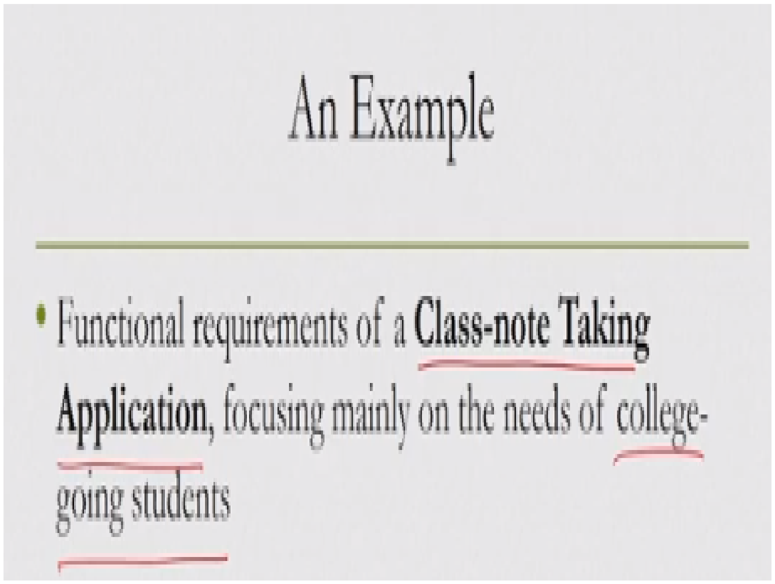
SRS

- Hierarchical structure (R1, R1, R1.1 etc)
- Each R (e.g, R1, R1.1 etc) represents a function with clearly defined input and output and some description (optional)

Now in SRS there are specific formats to be used generally what is followed is a hierarchical structure with some name given to the hierarchy levels like r 1, r 1.1 and so on. And at each level or r where r 1.1 etcetera which indicates the hierarchy each level represents a function with clearly defined input and output and some description which is optional. So when we are trying to specify SRS essentially we need to use these notations that; is a common practice that we create hierarchy label.

The hierarchies each level of the hierarchy should be given a label with some name and some description and these levels can further be decomposed into sub levels with similar notation. Now let us try to understand this idea of creating SRS from a functional requirement hierarchy let us try to understand this with respect to one example.

(Refer Slide Time: 18:33)

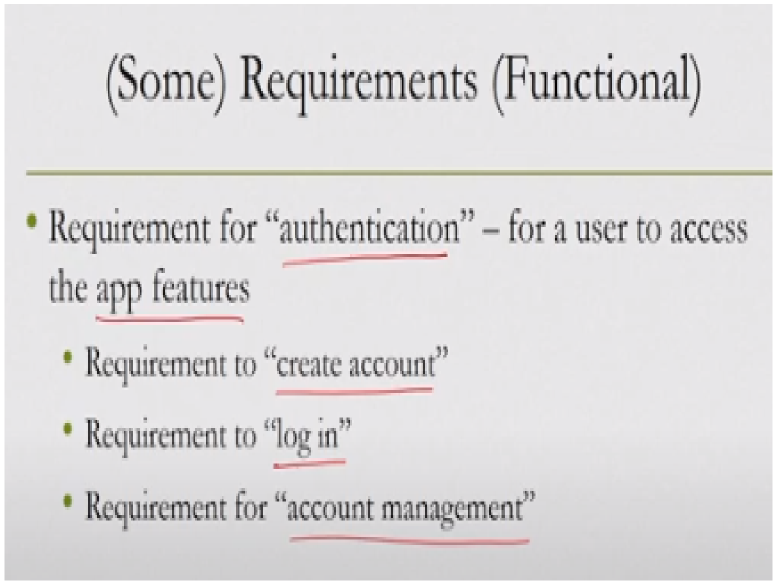


An Example

- Functional requirements of a **Class-note Taking Application**, focusing mainly on the needs of college-going students

If you may recollect we introduced this example in one of the earlier lectures this is a class note taking application where the primary users are likely to be the college going students. So earlier we have seen how to create usability requirements for this application using the contextual inquiry method. Now let us try to see how we can create a functional requirement specification for this application.

(Refer Slide Time: 19:06)



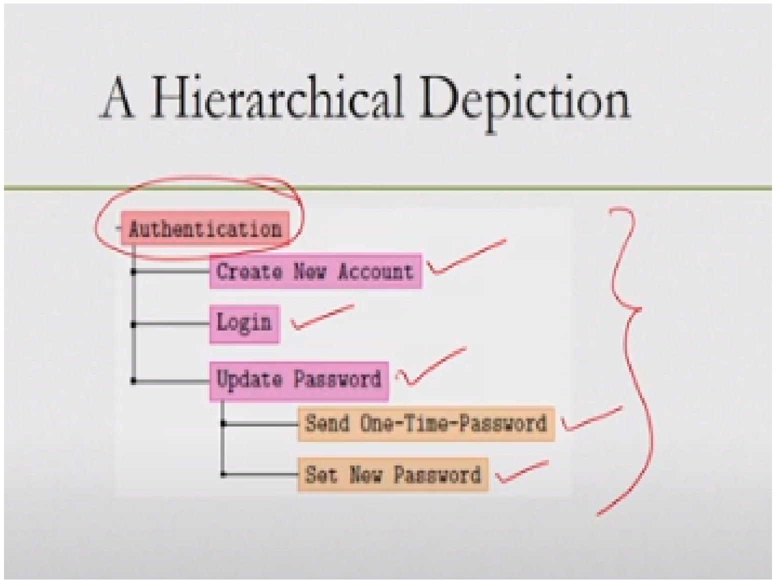
(Some) Requirements (Functional)

- Requirement for "authentication" – for a user to access the app features
  - Requirement to "create account"
  - Requirement to "log in"
  - Requirement for "account management"

What are the requirements that we can identify for this application? One obvious requirement can be authentication that is for a user to access the app feature so the user needs to authenticate which is common for all the apps. So we can have such a function as a requirement authentication function now authentication function can further be decomposed into sub functions.

For example there can be one requirement to create an account that can be one sub function under authentication. Then there can be a requirement to login that can be another function under authentication function one more function can be management of the account, account management it can be treated as another sub function of the authentication function.

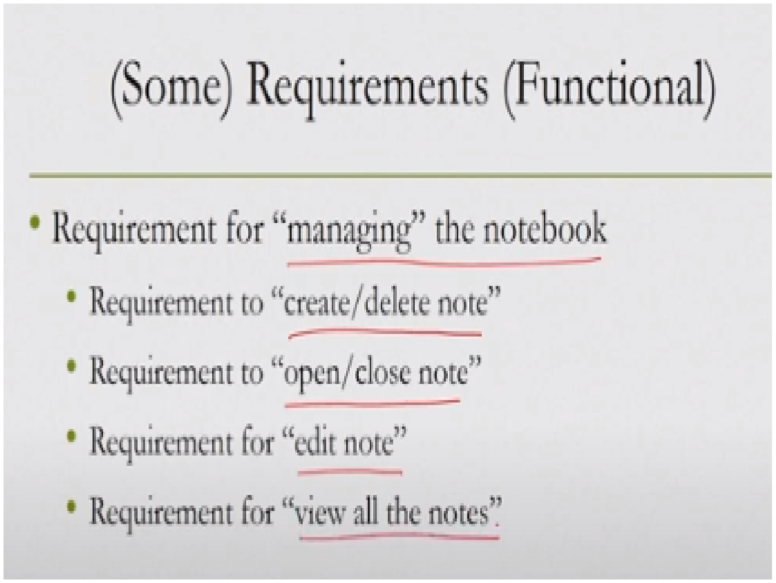
(Refer Slide Time: 20:13)


A Hierarchical Depiction

So diagrammatically we can put these functions and sub functions in the form of a hierarchy as shown in this figure. So here at the top level of the hierarchy we have authentication function then under authentication we have these 3 sub functions create new account login and update password. So update password can be one sub function under authentication function.

Now update password can have sub functions under it as well for example there can be one sub function send one time password can be another sub function set new password. So this can be one way of creating a functional hierarchy for the authentication function. Of course it may be noted that this cannot be the only way possible you are free to create your own hierarchy for authentication you can add more functions edit some functions and so on. So this is just one example of how such a hierarchy can be created.

(Refer Slide Time: 21:29)

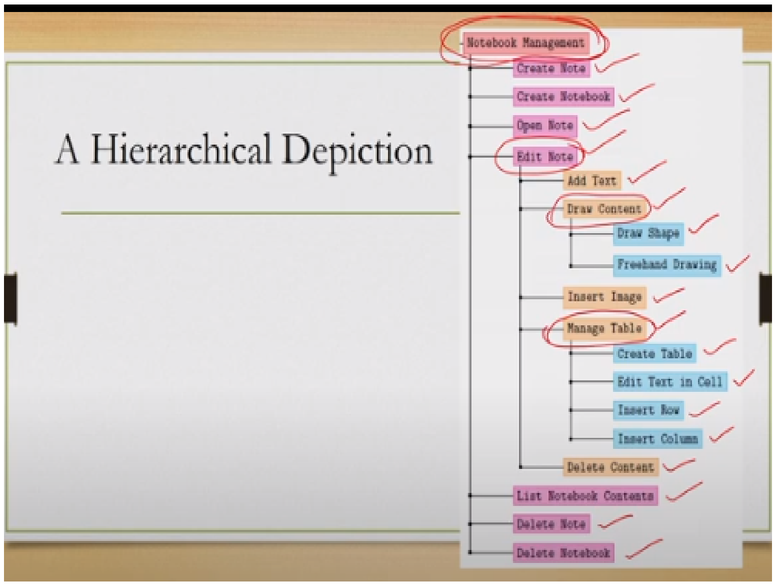


(Some) Requirements (Functional)

- Requirement for "managing" the notebook
  - Requirement to "create/delete note"
  - Requirement to "open/close note"
  - Requirement for "edit note"
  - Requirement for "view all the notes"

Let us look at another high level function this is related to managing the notebook in the application. So this can be a higher level functional requirement that the user wants to manage the notebook. Now under these functions there can function there can be many sub functions again it can lead to a hierarchy of functions and sub functions. For example there can be a requirement to create or delete note, can be a requirement to open or close a note. Can be a requirement for editing a note, can even be a requirement for viewing all the notes and so on.

(Refer Slide Time: 22:33)



A Hierarchical Depiction

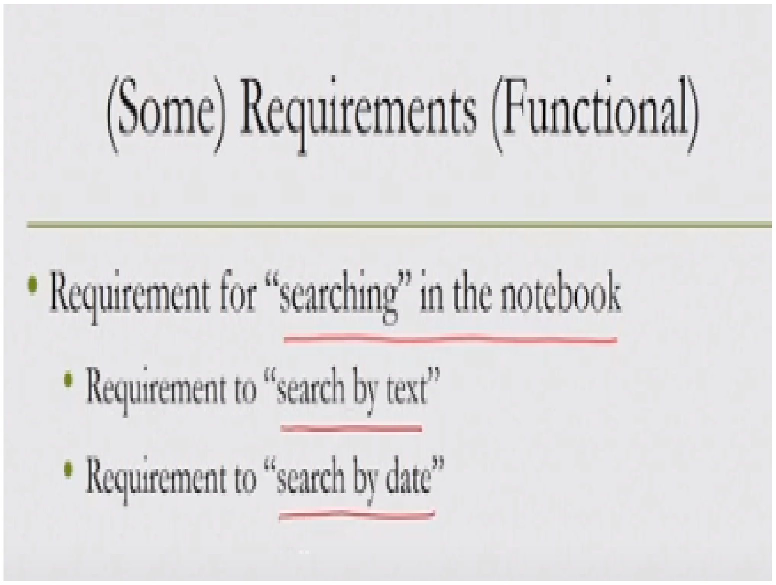
So we can create another hierarchy for notebook management function graphically we can have a representation like the one shown here for such a hierarchy. So at the top level we have notebook management functionality under this there are some functions create note, create notebook, open note, edit note, list notebook contents, delete note, delete notebook etcetera. Under edit note we can have further decomposition add text, draw content, insert image, manage table, delete content etcetera.

Under draw content further decomposition is possible like draw shape, freehand drawing etcetera similarly under manage table further decomposition is possible with sub functions like create table, edit text, insert row, insert column and so on. So what this hierarchy shows is that? It is possible to create a decomposed functional requirement hierarchy which will make it easier to understand the overall requirements.

Instead of this hierarchy if we have simply mentioned this top level hierarchy like notebook management then it would have been difficult to actually specify input output as well as description. What this is supposed to do? Because in that case it would have been very complex to specify those things instead of that by decomposing we are able to better manage this complexity by dividing it into sub function.

So then now what we need to do is basically specify only the input output for the sub functions which is much more easier compared to specifying everything for only one function.

(Refer Slide Time: 24:49)

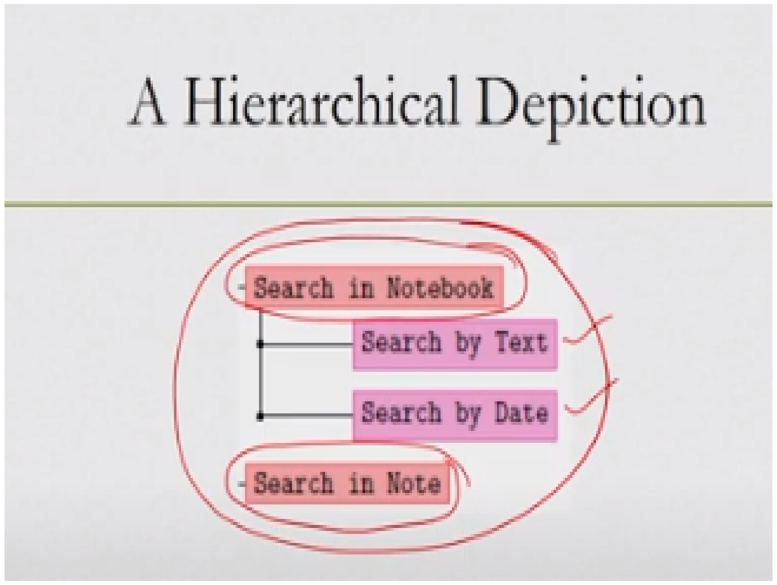


(Some) Requirements (Functional)

- Requirement for "searching" in the notebook
  - Requirement to "search by text"
  - Requirement to "search by date"

Let us see one more functional requirement example for the same application requirement for searching in the notebook. Now this can again be considered to be a higher level functional requirement top level and under this there can be several sub functions again creating a hierarchy. Requirement for searching by text this is one possible sub function another possible sub function can be searched by date.

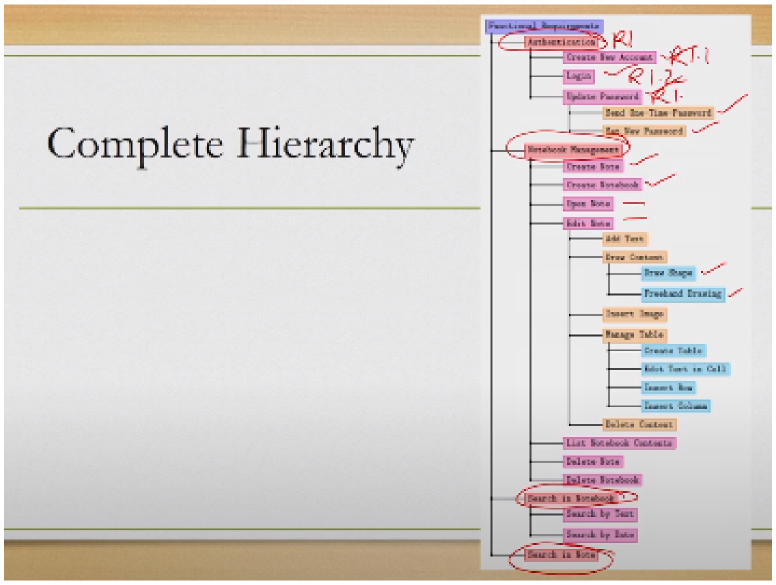
(Refer Slide Time: 25:24)



A Hierarchical Depiction

Then we can have a hierarchical depiction of the functional hierarchy in this form where we have search in notebook under which there are 2 sub functions search by text, search by date. Similarly we can have another high level hierarchy search in note and it is possible to decompose it further.

(Refer Slide Time: 25:55)

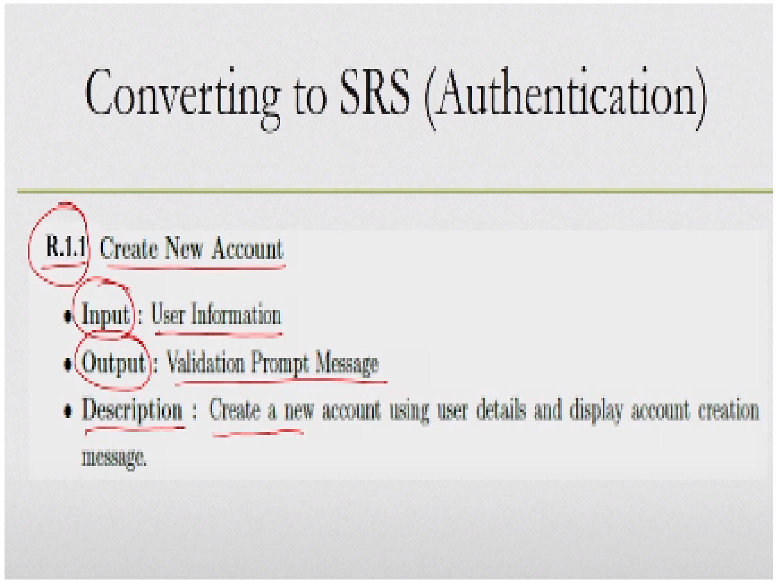


So then what we get what we get is a larger hierarchy so we merge these hierarchical descriptions together to get a larger hierarchy to represent the functional requirements of the system. Of course this is not complete but it will give you some idea of what we mean by hierarchy. So, here as you can see we have top level functional requirements authentication, notebook management, and search in notebook, searching note.

Then we have second level hierarchy create account login, update, password similarly create note, create notebook and so on. Then we can have third level also one time password, set new password even fourth level in this case draw shape, free and drawing etcetera. So there is no restriction on up to how much level we can go the more decomposition is the better for specification.

It will bring in clarity of thought if you have very less number of levels and everything in the single level then it is generally considered to be not a good specification assuming. That the system is complex for very simple system of course that multiple levels of hierarchy; may not be required.

(Refer Slide Time: 27:41)


Converting to SRS (Authentication)

R.1.1 Create New Account

- Input : User Information
- Output : Validation Prompt Message
- Description : Create a new account using user details and display account creation message.

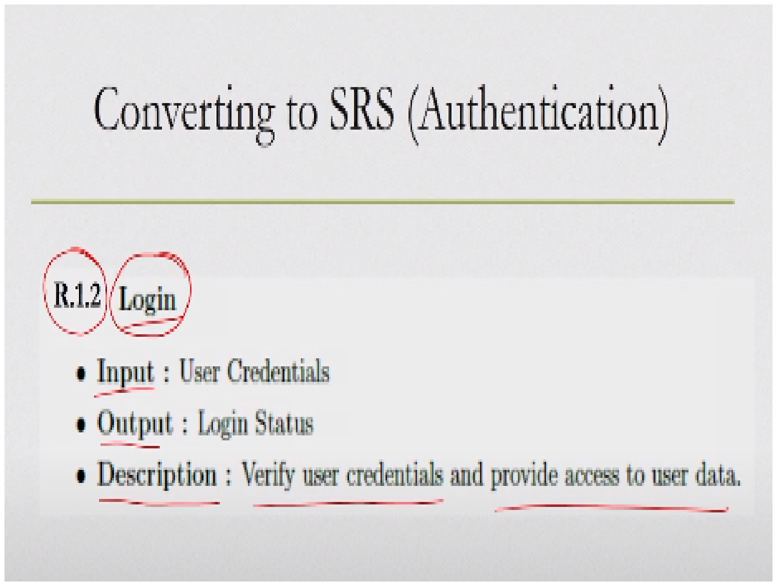
Now so far what we have discussed is just a visual depiction but that is not a specification as such for the requirements. Now we need to have a textual description as I mentioned before so textual description follows its own format with level numbers r 1, r 1.1 etcetera and function name input specification output specification and some function description. Still considering the function as a black box rather than describing the actual algorithm to implement the function.

For example let us consider this functional requirement create new account so this is as you can see it is second level. So we can call this level 1 and then create new account can be termed as level 1.1 the same thing we have used to denote in the textual description this is the level indicator then we have given it a name which is the same name that we have used earlier create new account.

Then there is one input specification so what input this function takes it simply mentioned user information. So it can be any information so it is possible to keep it vague at this stage because we are not going to discuss about how to implement? So the flexibility should remain output specification also is required here it is mentioned that some message will be given as output. And then there is this optional section on description of the function.

So it says what the function does purpose simply says create a new account using user details and display account creation message that is simply the purpose so in this way we can specify the functions in the functional hierarchy.

(Refer Slide Time: 30:00)



Converting to SRS (Authentication)

R.1.2 Login

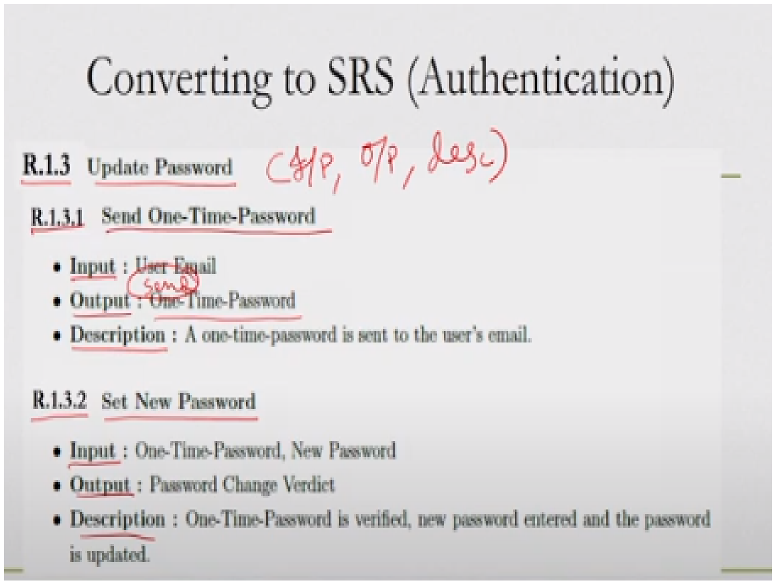
- Input : User Credentials
- Output : Login Status
- Description : Verify user credentials and provide access to user data.

Similarly we can create specifications for other functions say for example the second function is login so accordingly we gave it the number R 1.2 as you can see in the hierarchy. So the first one if it is given R 1.1 then we can call it R 1.2 then R 1.3 then this 1.3.1 and so on. So we are following this nomenclature note that this need not be the only way to represent levels and in different places you may find some other ways to represent these levels of the hierarchy.

However here in this course we will follow this convention so the level number is followed by the name of the function some name you have to give, then input specification, output specification and description brief description of what the function does? Like it simply says verify user credentials and provides access to user data. Note that nowhere we are referring to any details related to how the function can be implemented.

For example here in login we are not telling how this function will produce the output with from the input. So how this processing takes place or the algorithm to create the output is not specified anywhere it only states the purpose rather than the actual way to implement it.

(Refer Slide Time: 31:48)



Converting to SRS (Authentication)

R.1.3 Update Password (I/P, O/P, desc)

R.1.3.1 Send One-Time-Password

- **Input :** User Email
- **Output :** send One-Time-Password
- **Description :** A one-time-password is sent to the user's email.

R.1.3.2 Set New Password

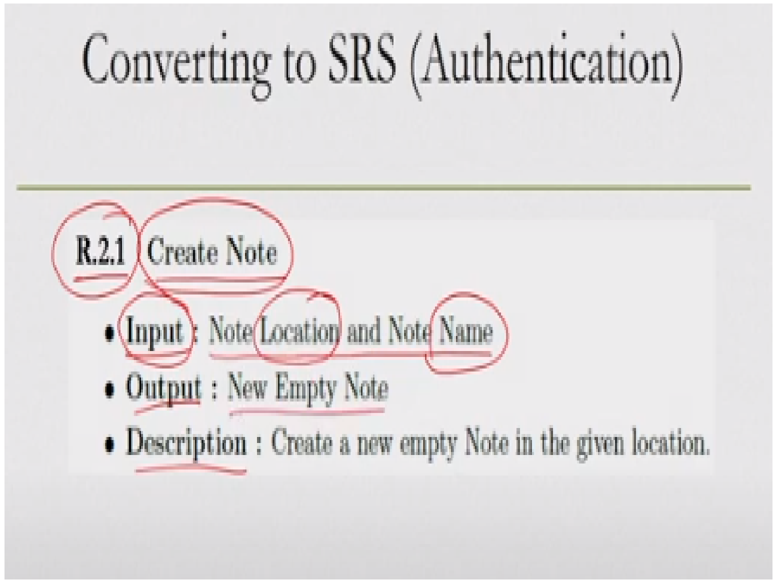
- **Input :** One-Time-Password, New Password
- **Output :** Password Change Verdict
- **Description :** One-Time-Password is verified, new password entered and the password is updated.

Some more examples R 1.3 update password then 1.3.1 send one time password input, output and description. It is generally preferable that higher level also you specify the input, output and description optionally for each level function you should do that not only for the leaf level notes that are mentioned here. So for our 1.3 also you should specify what is the input? What is the output and what is the description?

Then do the same for R 1.3.1 that is the sub function of 1.3 as well as our 1.3.2 this is the another sub function of R 1.3 set new password input output description. So at each level for every function you should ideally specify the input output and description for each and every function in that level. One common mistake occasionally we perform is that when we are specifying the output we also specify some functional words.

For example here send one time password output we may be tempted to write send one time password. Now the with the addition of the word send it no longer remains an output rather it becomes a process in itself so try to avoid using this type of action words in either input or output if you are using action words then the description is wrong. So you must keep this in mind and avoid using action words any sort of action words in the input and output. Otherwise those will not remain input and output instead they will become functions itself.

(Refer Slide Time: 33:59)


Converting to SRS (Authentication)

R.2.1 Create Note

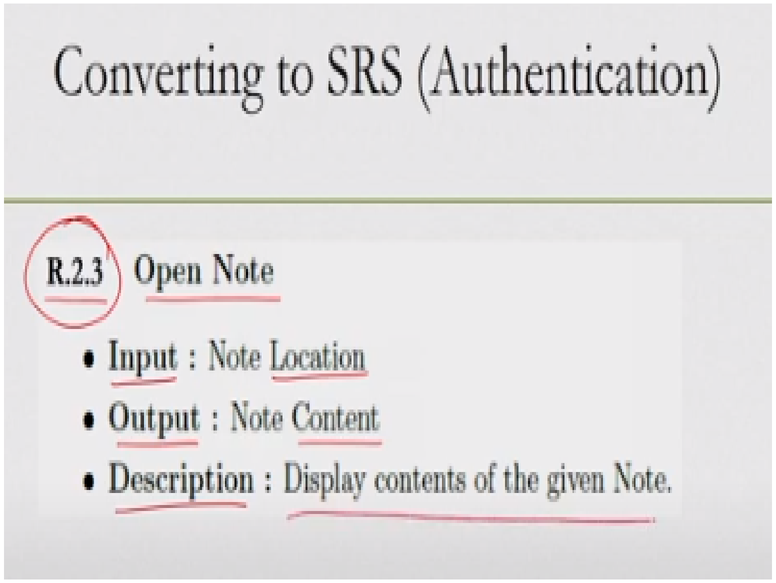
- **Input** : Note Location and Note Name
- **Output** : New Empty Note
- **Description** : Create a new empty Note in the given location.

So another top level note create note again input output description specification is given in a similar manner. So you can now get some idea of what we mean by formal specification. So here we are giving a level name**,** a label to the level name, name of the functional requirement some idea of the input to be provided to the function in a very flexible and open manner. It should not be very specific because here still we are talking of black boxes as functions so we are not going into the details of any algorithmic design.

So our input should be kept as much open as possible like here we are simply mentioning that provide as input location and name but in which format how all these details we are not providing so that is a flexibility that we want to keep at this stage. Similarly for output we are not specifying the exact nature of the output instead a broad idea of what the output should look like and what it should what purpose it should solve rather than exactly which format it should be produced in and how it should look.

So this idea of labels, names input, output specifications. As well as the idea to keep the inputs and outputs flexible and open to interpretation for subsequent design and implementation stages are key things that should be remembered while designing and specifying the functions.

(Refer Slide Time: 36:04)



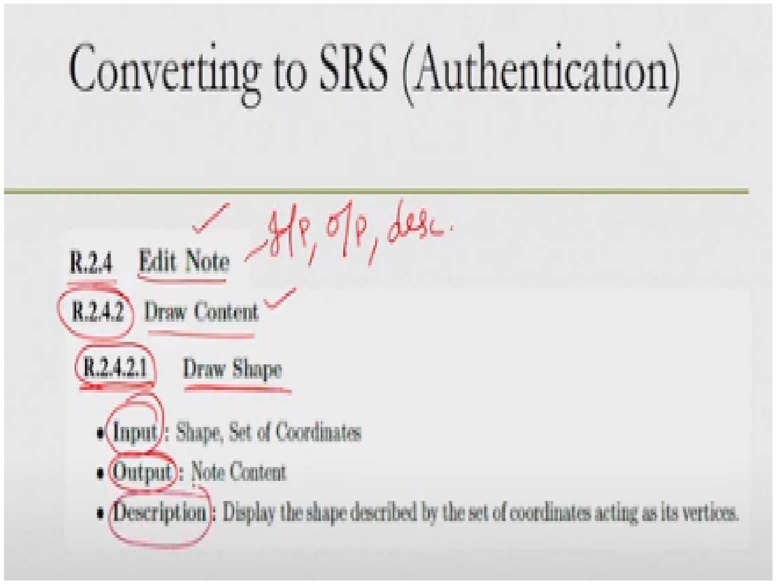
Converting to SRS (Authentication)

R.2.3 Open Note

- Input : Note Location
- Output : Note Content
- Description : Display contents of the given Note.

Then comes open note R 2.3 again with input, output and description and this openness is here also note location but not exactly in which manner whether it is a single number, coordinate that we are not specifying. Similarly content what this content means it is left to be decided in later stages rather than specified here itself and also a description broad description of what the function is supposed to do?

(Refer Slide Time: 36:42)



Converting to SRS (Authentication)

R.2.4 Edit Note ✓ – I/P, O/P, desc.

R.2.4.2 Draw Content ✓

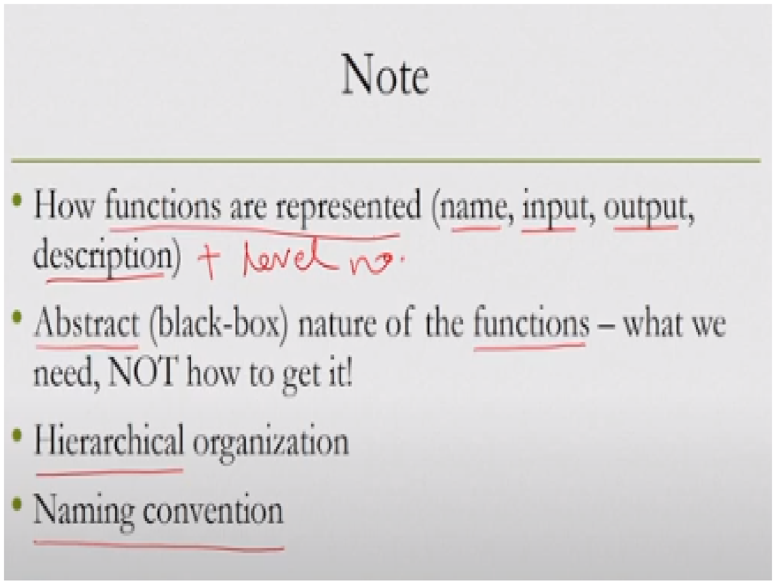
R.2.4.2.1 Draw Shape

- Input : Shape, Set of Coordinates
- Output : Note Content
- Description : Display the shape described by the set of coordinates acting as its vertices.

In this way we can specify all the functions in the hierarchy for simplicity and brevity we have omitted certain parts. Like here R 2.4 edit note the input, output and descriptions are omitted only the lowest level note shown to have this description. But remember to add it for each level like R 2.4, R 2.4.2 draw content as well as 2.4.2.1 draw shape which is a sub function of this function which is in itself; as a function of this function.

So draw shape is a sub function of draw content, draw content is a sub function of edit note and for each of these levels we should have input, output and description as shown here input here output here and description. So when we have created this hierarchy our next task is to create this textual description in the format shown in the examples. Again this is not the only way to specify there can be a different; conventions followed however broadly it should convey the message of the level which level the function is situated at then the function name, input, output and description.

(Refer Slide Time: 38:17)



Note

- How functions are represented (name, input, output, description) + level no.
- Abstract (black-box) nature of the functions – what we need, NOT how to get it!
- Hierarchical organization
- Naming convention

So you should note in the examples how the functions are represented name, input, output, description, plus the level number all 5 are important. Then another thing to be noted here is that the abstract nature of the representation. So none of the functions that we have seen in the examples tell us anything about how to implement those how to convert them to notes? So effectively what we are looking at is a black box where we provide the input and get the output without bothering about how we get the output.

So that abstract nature of specification should be noted and the hierarchy of course how we created the hierarchy to represent the requirements should be noted and should be followed in problems that you face. Also as I said follow the naming conventions typically what we have shown here can be followed but there can be other possibilities.

However anyone should be followed consistently throughout the life cycle. So we have shown one convention where we gave particular level of the hierarchy and used name for the function then input kept it open, output kept the scope open, some optional description part which should be done for each and every function in each and every level. It should not be considered only required at the last level even at the higher levels also for each function you should have ideally input, output and description specified.

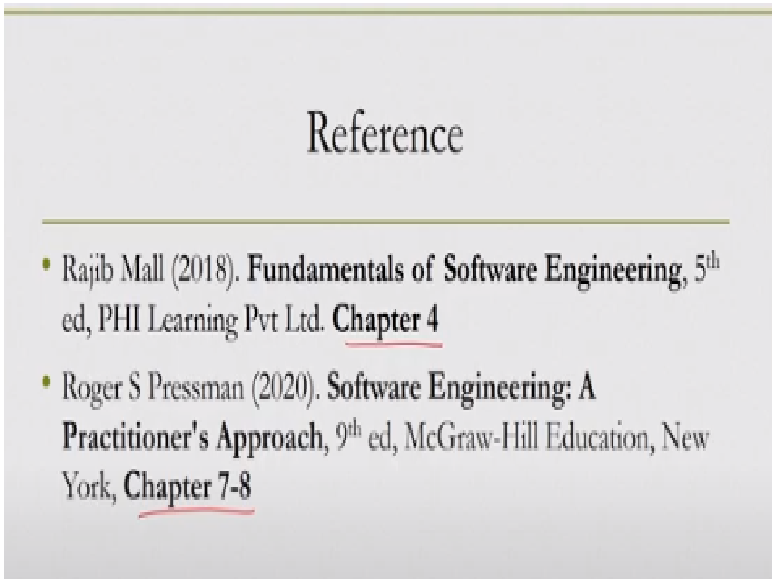
(Refer Slide Time: 40:16)

Take Home Exercise

- Complete the SRS!

The example that I have shown of course is not complete and as a put forth thought you can think of completing the hierarchy as well as the specification.

(Refer Slide Time: 40:30)



Reference

- Rajib Mall (2018). **Fundamentals of Software Engineering**, 5th ed, PHI Learning Pvt Ltd. **Chapter 4**
- Roger S Pressman (2020). **Software Engineering: A Practitioner's Approach**, 9th ed, McGraw-Hill Education, New York, **Chapter 7-8**

So with that we come to the conclusion of this topic so what we have covered today is what are functional requirements? What are the things we should keep in mind while creating functional requirements and how to specify those requirements? So one thing you should keep in mind is that this, functional requirements need not be created based on end user studies. Some domain knowledge plus some discussion with the clients may be good starting point to create the functional requirements.

Then subsequently once usability requirements are available some of those requirements can be converted to functions and put in the functional requirement hierarchy and subsequently included in the SRS document. We shall see in a subsequent lecture how to do that, apart from that other things we should keep in mind is that how the functions are specified? How they are decomposed how they are labeled and what are the things to be specified in the SS?

To recollect we need to specify for each function at each level its level, its name input, output and purpose. And in the input and output we should not be too specific about the nature of the input and output rather we should keep it open for interpretation at later stages of the development life cycle. Because here we are not bothered about how the functions are to be implemented rather what the function should do?

Accordingly we can keep the input and output somewhat open ended and flexible to interpretation whatever I have discussed today can be found in these books for further details you can refer to chapter 4 of the first book and chapter 7 to 8 of the second book. With that we conclude today is lecture I hope you have got some idea of what we mean by functional requirement and how to represent it both visually as well as formally in textual format in the form of SRS document.

In subsequent lectures we shall see how usability requirements can be incorporated in the form of functions in functional requirements. So that our systems can have better usability that is all for today see you in the next lecture thank you and goodbye.