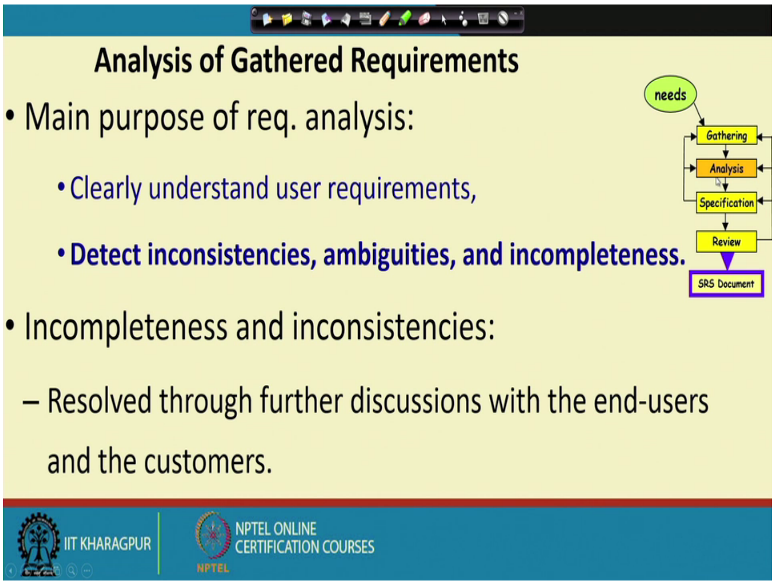
Software Engineering
Prof. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 16
Requirement gathering and analysis

Welcome to this lecture. In the last lecture, we were discussing about how to go about doing Requirements analysis and specification. We identified the main task that are to be completed during requirement analysis and specification said that first step to do the requirements gathering and then, need to analyze the gather requirements. And finally, go about doing the specification and then get the specification reviewed by the customer and other stakeholders.

Now, let us look at the second activity in the development process which is the analysis. We had looked at the gathering activity, seen the main activities that are to be performed during the requirements gathering and then, we are looked at a case study; simple case study to understand that what are the activities and how they are to be performed. Now, let us look at analysis of the gathered requirements.

(Refer Slide Time: 01:30)



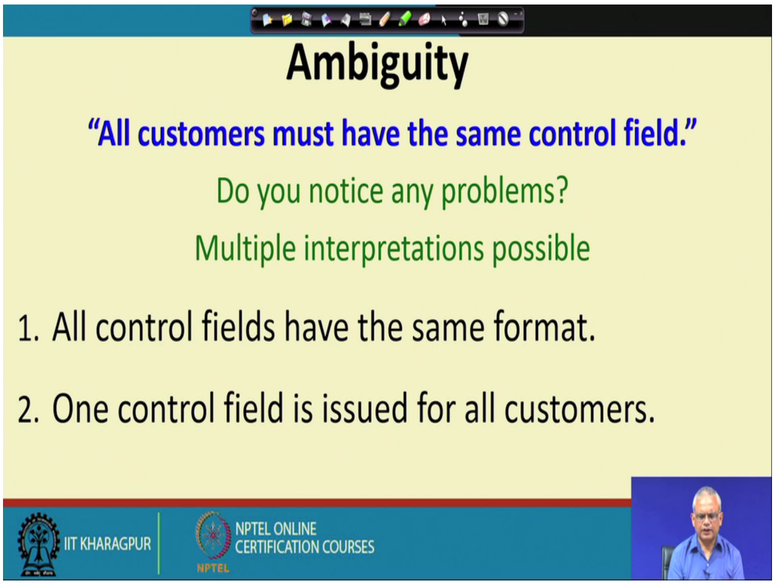
Analysis of Gathered Requirements

- Main purpose of req. analysis:
  - Clearly understand user requirements,
  - **Detect inconsistencies, ambiguities, and incompleteness.**
- Incompleteness and inconsistencies:
  - Resolved through further discussions with the end-users and the customers.

So, this is the second task that we have written here. In the requirements process, the main focus analysis is to identify the requirements problems. For this, we have to go through all the gathered requirements and clearly understand what all is required by the customer and then, we analyze to identify and remove requirements problems. There are 3 main problems that typically exist in a requirement gathered requirements; these are called as inconsistencies, ambiguity, and incompleteness.

Let us try to find out that what are these three problems inconsistency, ambiguity and incompleteness and how to identify these problems and once the problems are identified these are eliminated by discussing with the customer end user to remove this problems.

(Refer Slide Time: 02:57)



Ambiguity

"All customers must have the same control field."

Do you notice any problems?

Multiple interpretations possible

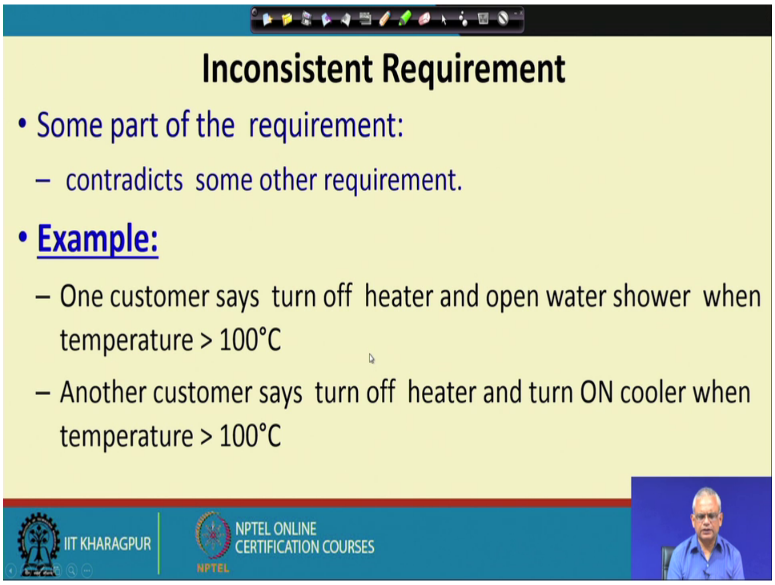
1. All control fields have the same format.
2. One control field is issued for all customers.

Now, first let us look at the ambiguity problem. Ambiguity as the name says is that the requirement is beg; it is not precise enough, it can have multiple meanings. Let us start by taking an example. Let us say one of the requirements that was obtained is that all the end users are customers, have the same control field. This was one of the statement made this is the actually an ambiguous requirement all customers must have the same control field.

What are the problems here? The problem is that different interpretations are possible. Does it mean that the control field is one control field which is shared with the multiple customers and if the change it, other customers are able to cheat and so on or is it that similar control fields are present in with different customers?

We need to discuss with the customer to check what was really meant is; is that there is only one cause control field which is shared and as the action done by each controller each customer it also affects the other customers or is it that they have similar control fields here same control field? But then, these are independent.

(Refer Slide Time: 05:08)



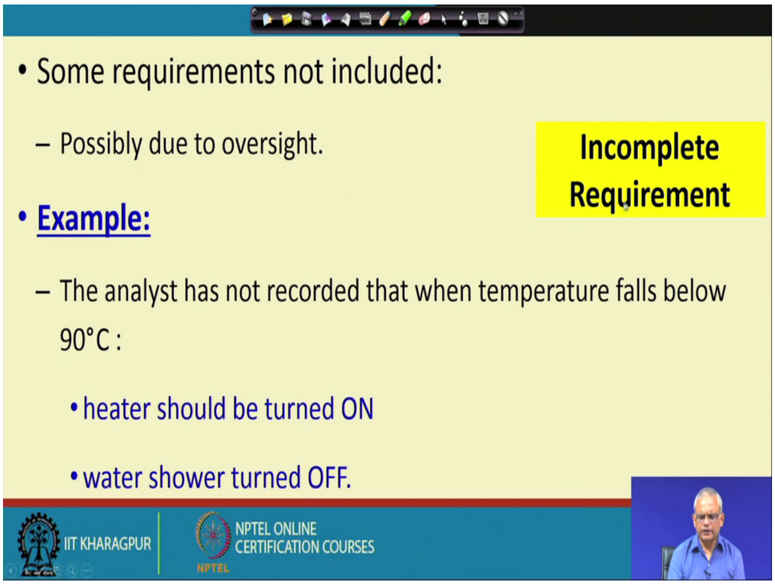
Inconsistent Requirement

- Some part of the requirement:
  - contradicts some other requirement.
- **Example:**
  - One customer says turn off heater and open water shower when temperature > 100°C
  - Another customer says turn off heater and turn ON cooler when temperature > 100°C

Now, let us look at an example of a inconsistent requirement. As the name says in the inconsistent requirement, some requirement contradicts some other requirement. For example, let us say one customer says that turn on the heater and open water shower when the temperature and the chemical reactor exceeds 100 degree centigrade.

But then, another customer has given the requirement saying turn off the heater and turn on the cooler; not the water shower. So, the same requirement when temperature exceeds 100 degree centigrade, the action to be taken is contradictory; need to discuss with the customer which a which one is the correct.

(Refer Slide Time: 06:12)



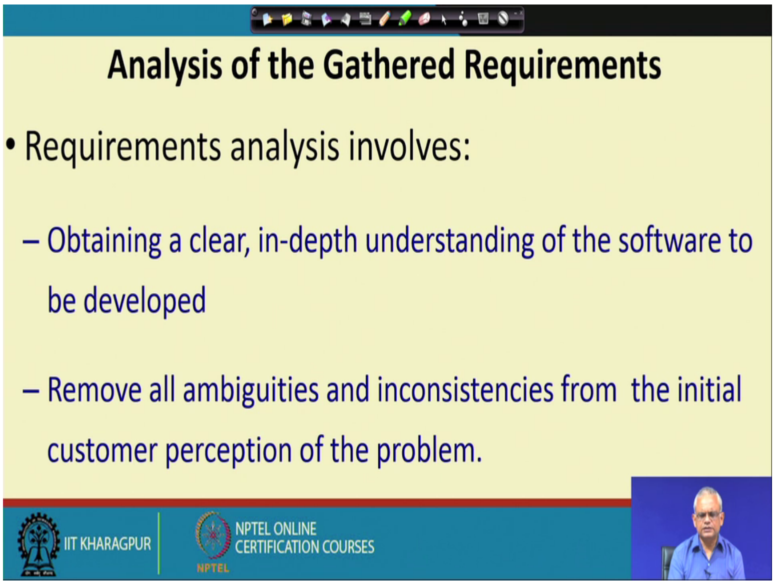
- Some requirements not included:
  - Possibly due to oversight.
- **Example:**
  - The analyst has not recorded that when temperature falls below 90°C :
    - heater should be turned ON
    - water shower turned OFF.

Incomplete Requirement

Incomplete requirement is the third category of the problems that we need to the watch full or during analysis we need to identify all incomplete requirements and eliminate them. Incomplete requirements are the requirements that the customer has forgot or does not realize that these will be needed. Just to example; just to give an example and the customer has not recorded or has not expressed what will happen, when the temperature falls below 90 degree centigrade? They are says that when the temperature is greater than 100 degree centigrade do something; but then, on the lower side of the temperature has not recorded any action to be taken.

So, that say incorrect requirement; it is actually incomplete. We should have actually recorded that when the temperature is less than 90 degree; heater should be again turned on and the water shower should be turned off to maintain the temperature between 90 and 100.

(Refer Slide Time: 07:33)

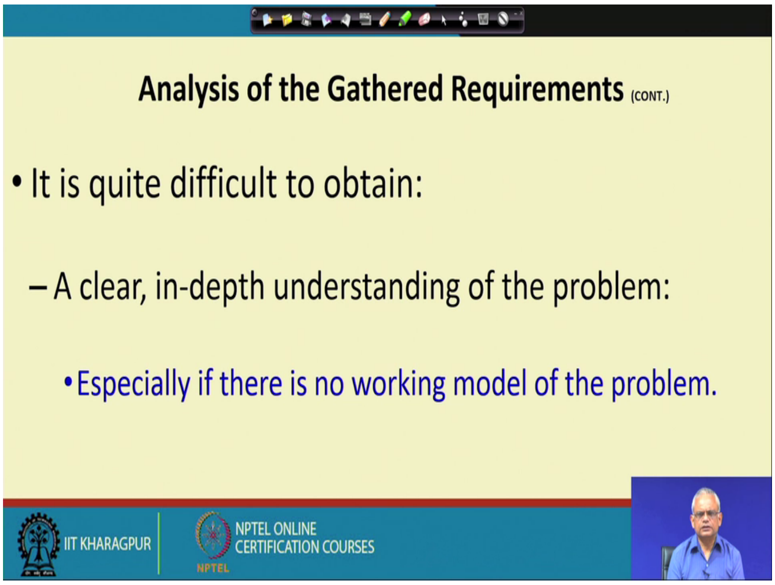


Analysis of the Gathered Requirements

- Requirements analysis involves:
  - Obtaining a clear, in-depth understanding of the software to be developed
  - Remove all ambiguities and inconsistencies from the initial customer perception of the problem.

During the analysis of the gathered requirement, must understand the gathered requirements well and then remove the ambiguities inconsistencies and incompleteness from the gathered requirements.

(Refer Slide Time: 07:53)

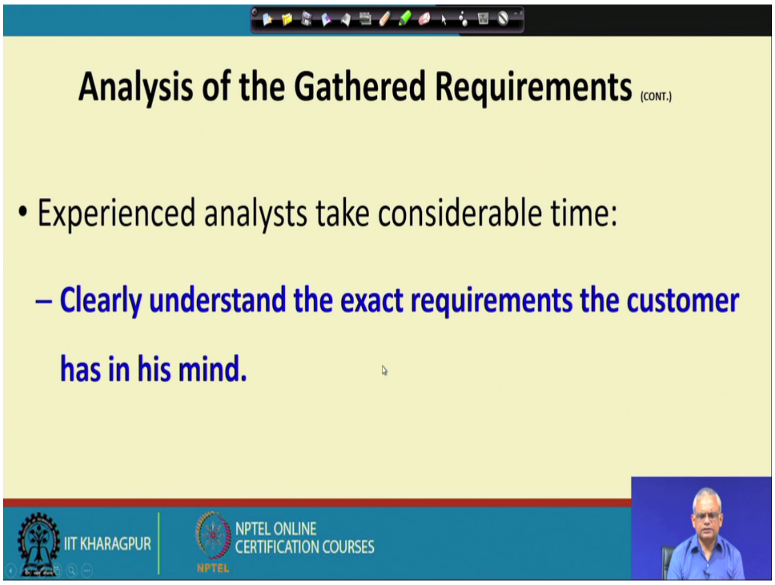


Analysis of the Gathered Requirements (CONT.)

- It is quite difficult to obtain:
  - A clear, in-depth understanding of the problem:
    - Especially if there is no working model of the problem.

This is even though we just said it simply, but it is actually a difficult task; without a working system to have a in depth understanding of the problem based on what the customers have in their mind is a difficult problem.

(Refer Slide Time: 08:15)

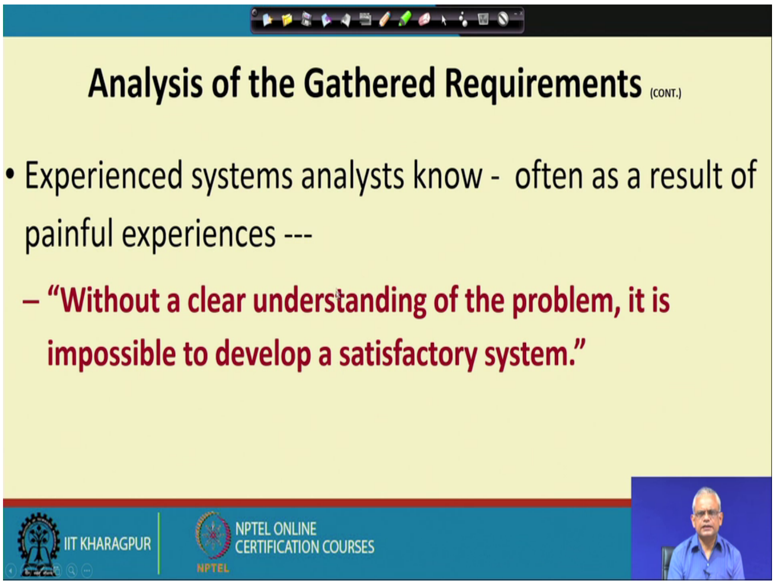


Analysis of the Gathered Requirements (CONT.)

- Experienced analysts take considerable time:
  - Clearly understand the exact requirements the customer has in his mind.

The experienced analysts that is the person who gathers requirement and specifies that typically called as analysts; the analysts take time to completely understand what the customers need do the analysis and the specify they devote lot of time here because this is a difficult task.

(Refer Slide Time: 08:50)

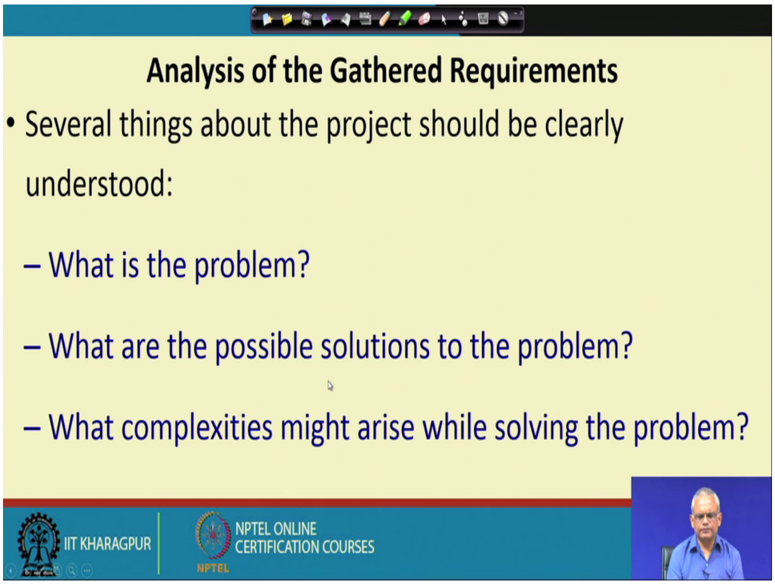


Analysis of the Gathered Requirements (CONT.)

- Experienced systems analysts know - often as a result of painful experiences ---
  - "Without a clear understanding of the problem, it is impossible to develop a satisfactory system."

The experienced analyst known that it is worth while spending little bit more time in doing the requirements gathering and analysis, work well. Because unless we are have a clear understanding of what is to be done; it is impossible to develop a satisfactory solution.

(Refer Slide Time: 09:18)

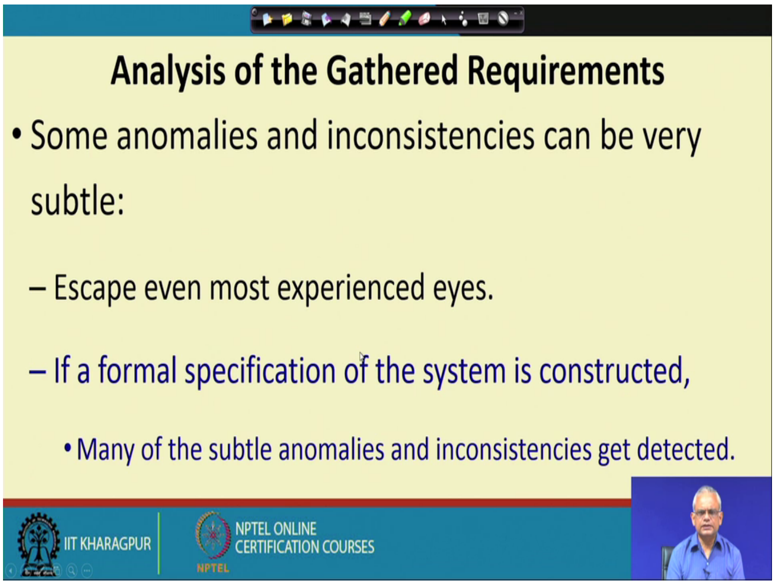


Analysis of the Gathered Requirements

- Several things about the project should be clearly understood:
  - What is the problem?
  - What are the possible solutions to the problem?
  - What complexities might arise while solving the problem?

During the gathering of requirements and the analysis, several things must be clearly understood that what are the problems that the user wants to solve; what features are needed and how can this be implemented? Because if the user wants to do something very far fetch, it is difficult to implement. It should not really accept that; we should change modifications to that which can be meaningfully solved and then, must understand what are the complexities while solving the problem.

(Refer Slide Time: 10:07)

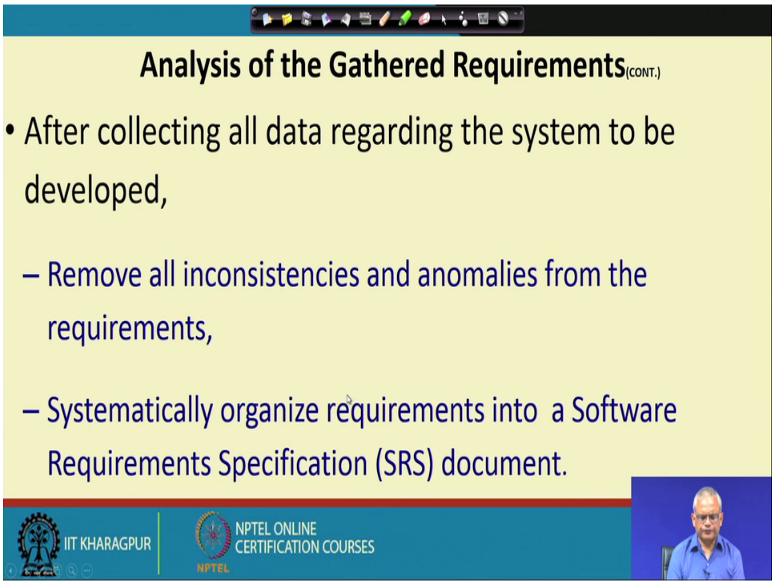


Analysis of the Gathered Requirements

- Some anomalies and inconsistencies can be very subtle:
  - Escape even most experienced eyes.
  - If a formal specification of the system is constructed,
    - Many of the subtle anomalies and inconsistencies get detected.

Even though, the analysis activities carried out very thoroughly, still some problems may remain and for this formal specification technique were very critical software can be develop because formal specification is a very expensive, time consuming task and if some part of the software extremely is sensitive critical parts can be we can have a formal specification after we have develop the requirement specification document.

(Refer Slide Time: 10:48)

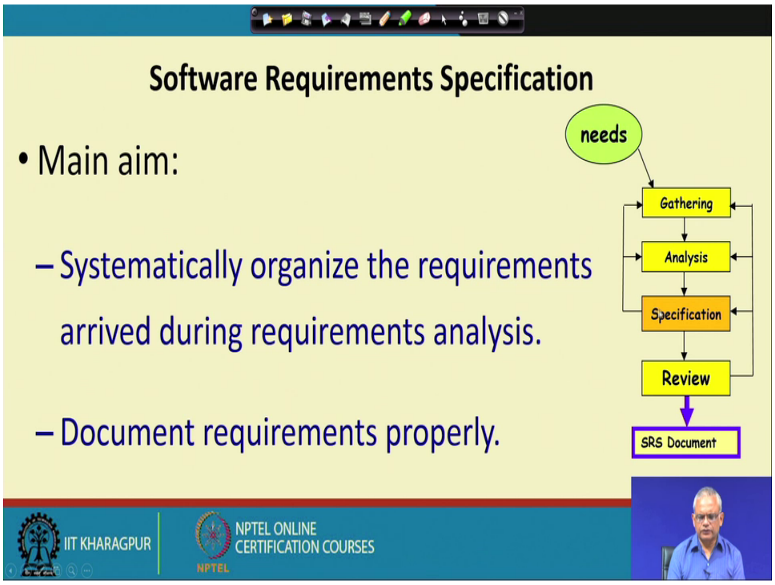


Analysis of the Gathered Requirements(CONT.)

- After collecting all data regarding the system to be developed,
  - Remove all inconsistencies and anomalies from the requirements,
  - Systematically organize requirements into a Software Requirements Specification (SRS) document.

Now, after we have removed all inconsistence anomalies incompleteness from the document, you go about documenting the requirements. Now, let us see how to write the SRS document is a very important skill. Let us see how to go about.

(Refer Slide Time: 11:15)

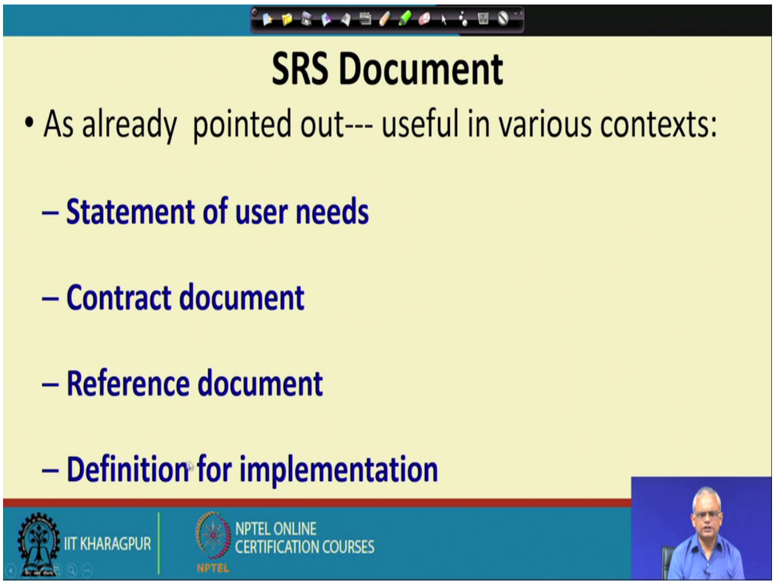


Software Requirements Specification

- Main aim:
  - Systematically organize the requirements arrived during requirements analysis.
  - Document requirements properly.

Now, let us look at the specification task that is how to write the requirements document. Here, we have the gathered requirements and all the problems from them have been removed through the analysis activity and now, we get down to systematically organize the requirements and then document them properly.

(Refer Slide Time: 11:44)

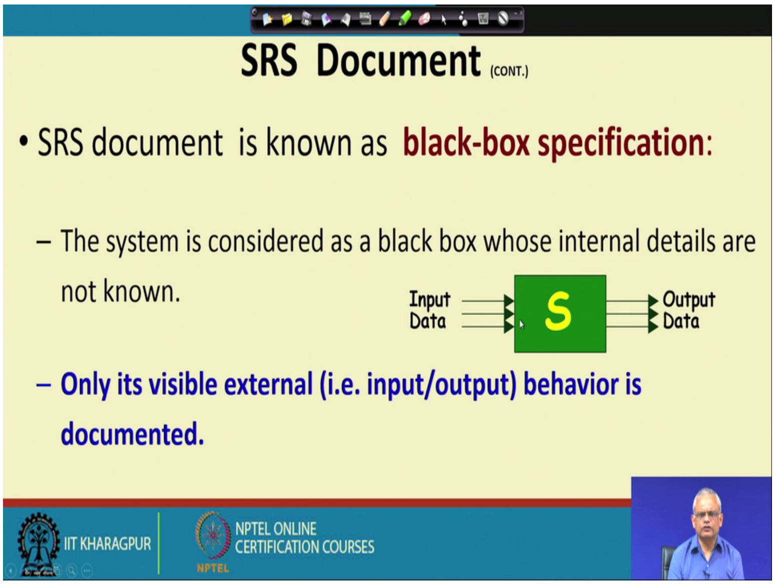


SRS Document

- As already pointed out--- useful in various contexts:
  - Statement of user needs
  - Contract document
  - Reference document
  - Definition for implementation

The SRS document is a very important document used by various types of users and our document should be written SRS document such that it can be understood and useful to different users because they may need look for different types of information in the SRS document. It serves as a statement of the user need. It is a contract document. It is a reference document for the development team to start development work; for the testers to design the test cases and also it is the definition of the problem based on which the developers start to develop the work.

(Refer Slide Time: 12:40)



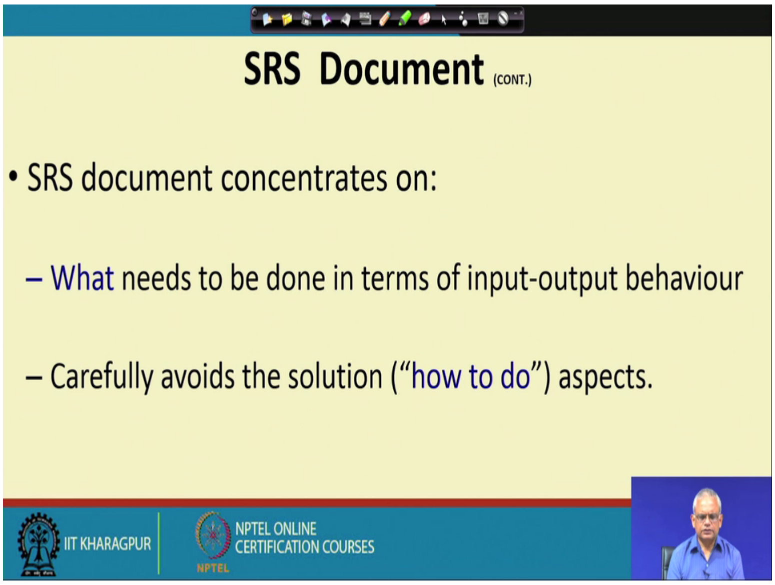
SRS Document (CONT.)

- SRS document is known as **black-box specification:**
  - The system is considered as a black box whose internal details are not known.
  - **Only its visible external (i.e. input/output) behavior is documented.**

The SRS document should be written in the form of a black-box specification. What do you mean by a black box specification of the problem? The black-box specification is actually the functionality as can be observed externally that is what is input to the system and what gets output to the by the system?

The system appears as a black box, we do not know that based on the input data which function here does what etcetera, etcetera. But we know that what will be input and based on this, what output will be generated that is the input output behavior. The internal details we should not document or try to document, we just document the input output behavior that is based on input what sort of output should be produced.

(Refer Slide Time: 13:58)

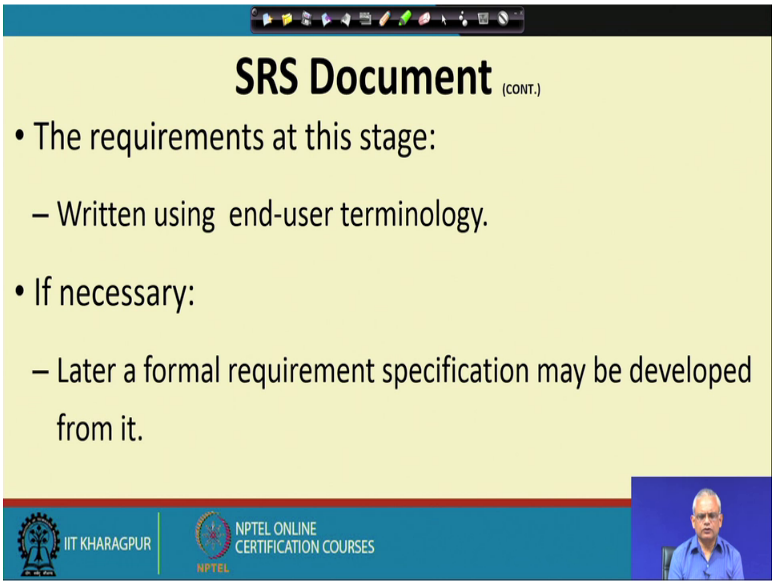


SRS Document (CONT.)

- SRS document concentrates on:
  - What needs to be done in terms of input-output behaviour
  - Carefully avoids the solution ("how to do") aspects.

In other words, you can say that the SRS document should focus on what needs to be done based on the input data, what the system should do rather than how to do.

(Refer Slide Time: 14:23)

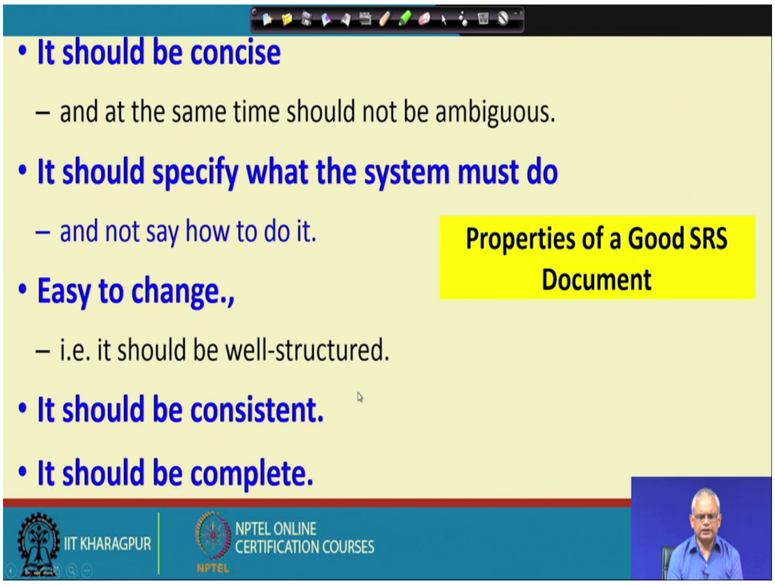


SRS Document (CONT.)

- The requirements at this stage:
  - Written using end-user terminology.
- If necessary:
  - Later a formal requirement specification may be developed from it.

The requirement document should be written in end user terminology because it is the end users who will see whether all the requirements have been satisfactorily represented. It is written as a document, which can be understood by several types of users that is the end user, the testers, the project manager, the development team members and so on. But then, if some parts are very critical separate formal requirement specification document can be written for that.

(Refer Slide Time: 15:15)



- It should be concise
  - and at the same time should not be ambiguous.
- It should specify what the system must do
  - and not say how to do it.
- Easy to change.,
  - i.e. it should be well-structured.
- It should be consistent.
- It should be complete.

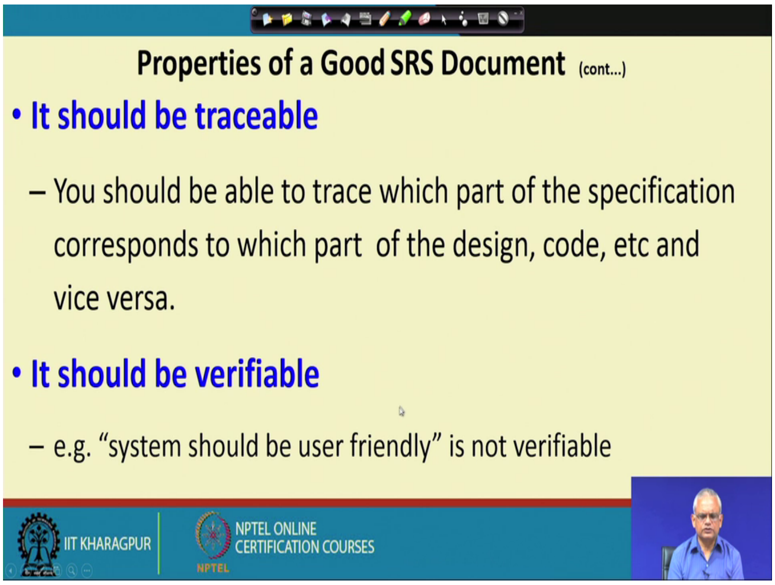
Properties of a Good SRS Document

Now, let us see before starting to write a document or what should be the sections, how should we go about writing the subsections and so on. Let us see overall what exactly is a good SRS document? One is that it should not be too (Refer Time: 15:37). It should be used the minimum writing that is needed should be concedes and at the same time should not leave out details are the ambiguous.

It should be a black-box specification; should specify what the system must do and not say how to do it. It should be easy to change; that is if some functionality tomorrow changes, we should be able to change very quickly and how do I we ensure that it is the document is easy to change is that we have to structure it well, like if you want to change one functionality, we should be able to change one plus should not have to change and multiple pages different places and so on.

It should be consistent. It is different functionalities, they should be consistent; should be complete, all functionality should have in represented the SRS document.

(Refer Slide Time: 16:50)



Properties of a Good SRS Document (cont...)

- It should be traceable
  - You should be able to trace which part of the specification corresponds to which part of the design, code, etc and vice versa.
- It should be verifiable
  - e.g. "system should be user friendly" is not verifiable

And also it should be traceable. Traceable means as it is functionality identified in the SRS document is implemented, we should be able to identify the functionality is implemented by which part of the design which part of the code and so on. To be able to have traceability, the first thing is that we have to a proper structuring of the document.

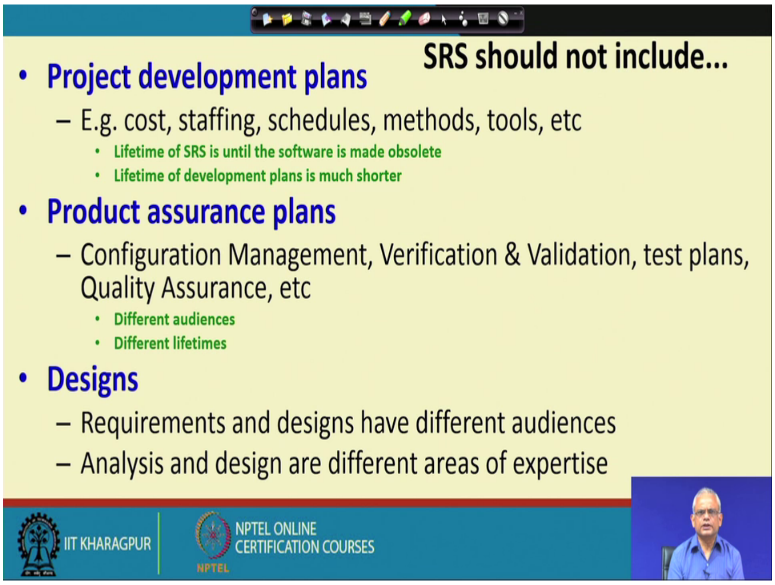
For example, the functions have to be precisely written each function should be identifiable by a number or something and should be written at one place and then, it should become possible that given a design element, we should be able to tell it implements which functionality or given a code segment that we want to change let us say or we find a error in a code segment, we should be able to tell which functionality is would have got affected.

So, this is called as traceability; very important concept that given a function requirement number should be able to tell which part of the design implements that and which part of the code implements that design. Also in the other way given a code element should be able to tell, it implements which part of the design and also it is functionality.

The SRS document should be written such that it is verifiable that is should be easily we can write test cases and check whether it can be we can clearly say whether the requirements is satisfied or not. If you have a requirement like let say the developed software should be user friendly. It is very difficult to test. Even if we test for user friendly, then it may not be accepted by all we might check that it is user friendly; but then others may not agree.

So, this is a very difficult requirement to verify while writing the requirement document we have to ensure that every requirement can be tested satisfactory.

(Refer Slide Time: 19:46)



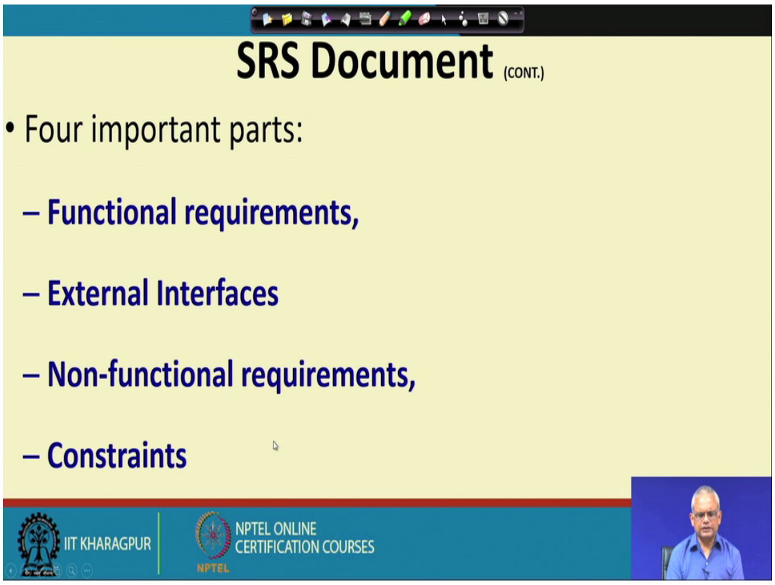
SRS should not include...

- Project development plans
  - E.g. cost, staffing, schedules, methods, tools, etc
    - Lifetime of SRS is until the software is made obsolete
    - Lifetime of development plans is much shorter
- Product assurance plans
  - Configuration Management, Verification & Validation, test plans, Quality Assurance, etc
    - Different audiences
    - Different lifetimes
- Designs
  - Requirements and designs have different audiences
  - Analysis and design are different areas of expertise

SRS document should not include not only the how to aspects; but other things like let us say project development plans like what will be the cost, staffing schedule, tools to be used etcetera because the SRS document is used even after the development work complete. So, the lifetime of the development plans are till the development completes; but the SRS document out lifts that and SRS document should not include development plans.

Similarly, it should not include testing assurance plans, Configuration Management, test plans, Quality Assurance because they have different audience and different lifetime. SRS document has several audience and we write will be those aspects which are if interest all the audience. The designs etcetera, they are also not to be included.

(Refer Slide Time: 21:04)



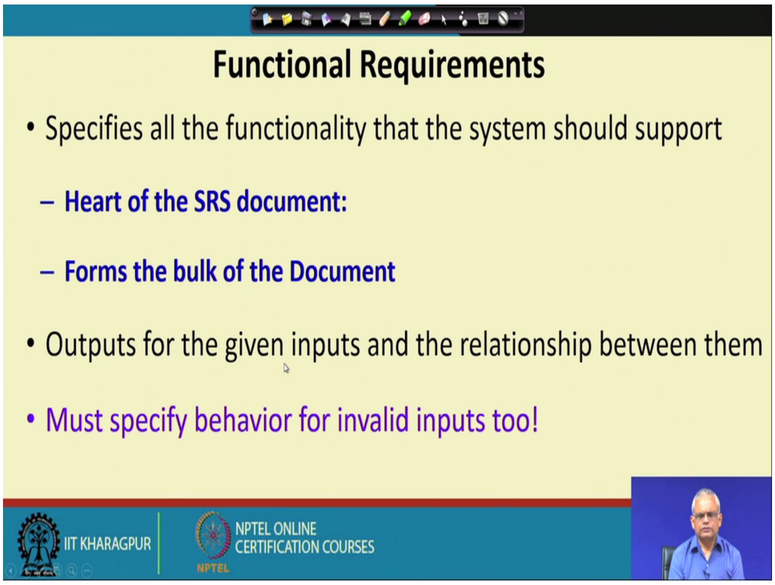
SRS Document (CONT.)

- Four important parts:
  - Functional requirements,
  - External Interfaces
  - Non-functional requirements,
  - Constraints

Now, let us see how to go about writing the SRS document. There are 4 important parts in any good SRS document. Possibly, the most important part the Functional requirements and then, you must identify the External Interfaces. The external software and hardware with which the software needs to interact and the users and then, the Non-functional requirements, we will see that there are some requirements which are not cannot be specified as functions.

We will see what is the distinction between a functional and non-functional requirement? We have to have a section on the non-functional requirements and what are various constraints on the system? For example, one constraint maybe that it has to follow certain standards or maybe that it has to use a open source database first.

(Refer Slide Time: 22:27)

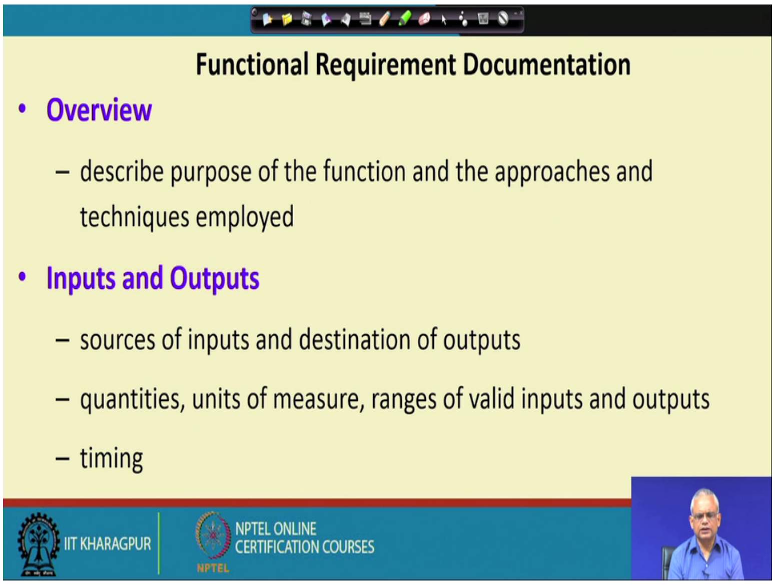


Functional Requirements

- Specifies all the functionality that the system should support
  - Heart of the SRS document:
  - Forms the bulk of the Document
- Outputs for the given inputs and the relationship between them
- Must specify behavior for invalid inputs too!

Let us concentrate on the functional requirements because this is the most important part of any SRS document and it forms the bulk of the document. Here, we identify what are the functionalities or facilities that the software will offer to the users and we document in the form of the input that the user will have to give; the output that will be produced and also we have to mention that what would happen when the user gives invalid input.

(Refer Slide Time: 23:16)



Functional Requirement Documentation

- Overview
  - describe purpose of the function and the approaches and techniques employed
- Inputs and Outputs
  - sources of inputs and destination of outputs
  - quantities, units of measure, ranges of valid inputs and outputs
  - timing

Identifying the functional requirements is a very important skill just to give an example, let us say give you a problem that you have to develop a library software.

So, what are the functionalities that it should have or let me give another problem that is you are asked to develop a word processing software that is somebody should be able to write a document, something like a Microsoft word or something that will be a good word processing software. So, what will be the functional requirements of that? To identify the functional requirements, we have to see what facilities it provides. For example, let us say library software. We have to look at who are the different users the librarian and then, the library members and possibly the account.

Now then, you have to look at what are the facilities required by each type of user. A library member might like to issue book, return book query whether book is available. The librarian might like to create members, it like to delete members, might like to check if there is fine against anybody, might like to create books, delete books etcetera. The account might like to check, what is the financial position, might like to enter the grants, might like to enter the fees, might like to register the expenses and so on.

So, from each users perspective, we have to look at the software terms of the facilities provided and each facility it basically some meaningful work that the user can accomplish using that and once you have identify the functional requirements each functional requirement, we need to document. For every function, we need to describe what will be achieved and what are the approaches and techniques that will be deployed and also need to identify the input and output. Typically, every function requires a input. It does some processing and produces some output. For example, query book function, for a library software. In the query book function, the user gives the name of the book. So, that is a input.

The software checks the book database for those books which manage the, which match the user query and then, it produces the output. So, every functional requirement must be documented in the form of a overview of what the functionality will achieve and then, it possible we need to identify the input and output.

For every functionality, it may not be feasible to produce to think of the input and output. We will look at more details of how to document the functional requirements; but the end of this lecture must be clear about what is a functional requirement? A functional requirement is a facility provided by the software using which user can get some meaningful piece of work done.

Let us now stop here and from this we will proceed; we will look at more details of functional requirements non-functional requirements, constraints and so on in the next lecture.

Thank you.