

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

**Lecture – 19**  
**Design Fundamentals**

Welcome to this lecture. In the last lecture, we are discussing about representing complex processing logic. If we write the complex processing logic in the form of a text description, then it becomes difficult for the developers to understand, for the testers to develop test cases. And also, it is very likely that in a text base description of a processing logic, certain combinations of combinations or variables and their corresponding action might get missed.

And therefore, it's a good idea that during requirement specification, if we find that the logic is likely complex; then we have to represent it in the form of a decision or a decision table. We had seen that these are very simple semiformal techniques and we can very easily develop the decision tree and decision table for any problem. Let's take an exercise.

(Refer Slide Time: 01:53)

**Exercise**

- Develop decision table for the following:
  - If the flight is more than half-full and ticket cost is more than Rs. 3000, free meals are served unless it is a domestic flight. The meals are charged on all domestic flights.

Flight D	Y	N	N	N
Ticket > 3000	–	N	Y	Y
Occupancy > 50%	–	–	N	Y
Free Meal	✓	✓	✓	✓
Charged Meal	✓	✓	✓	✓

The slide also features the IIT Kharagpur and NPTEL logos at the bottom, and a small video inset of the professor.

We want to develop decision table and decision tree for the problem given in slide. Just see here that the problem statement is very simple; but still the developers and testers might find it bit confusion. If the flight is more than half-full and the ticket cost in the

flight is more than 3000 rupees, then free meals are served. But in domestic flight, this rule does not apply and all meals are charged on all domestic flights.

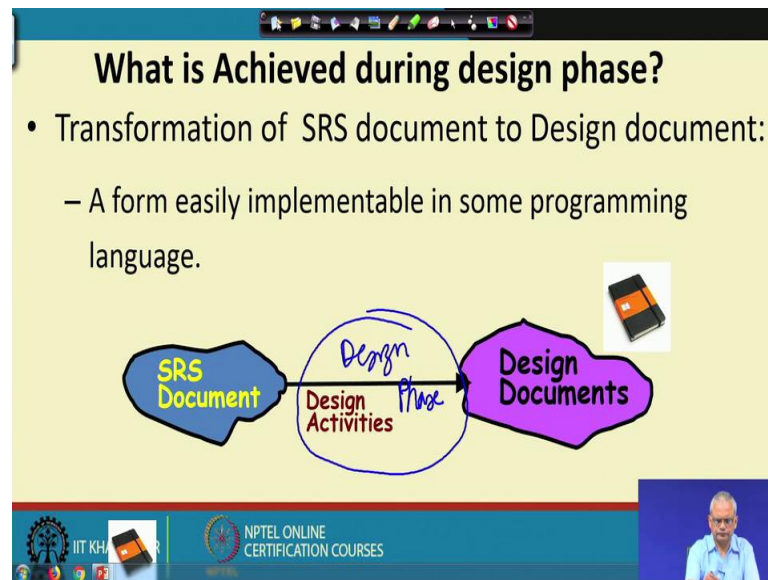
So, if you can see here that the one of the conditions here is that whether the flight is a domestic flight? So, our first condition is that 'the flight is a domestic? yes or no' and then another condition is that 'ticket cost: ticket cost greater than 3000 or not?' and then, we have to check whether 'the flight is half-full occupancy: greater than 50 percent'.

The actions are free meal or charge meal. If it is a domestic flight, then these we will have to charge the meal; but the flight is not domestic and the ticket cost is let us say less than 3000, then also it will be charged meal. If the flight is not a domestic flight and the ticket cost is more than 3000 and occupancy rate is less than 50 percent, then again, we charge the meal. But if it is not a domestic flight and the ticket cost is more than 3000 and the occupancy is also more than 50 percent then we give free meal. So, as you can see here that this gives a conceptually meaningful representation of this conditions.

But we can see here that these all are very simple condition that we are trying to represent here as an example. The conditions can become much more complex and therefore, the true use of a decision table can come out. Exercise to do: develop the decision tree representation for the same problem. Let's conclude our discussion on requirement specification and let's proceed to look at software design aspects.

Now, we will see some very basic issues in software design and then, we look at the procedural and object-oriented design principles. We will look at now software design and we will start with some very basic issues in software design. The designed activities are under taken once the requirement is complete.

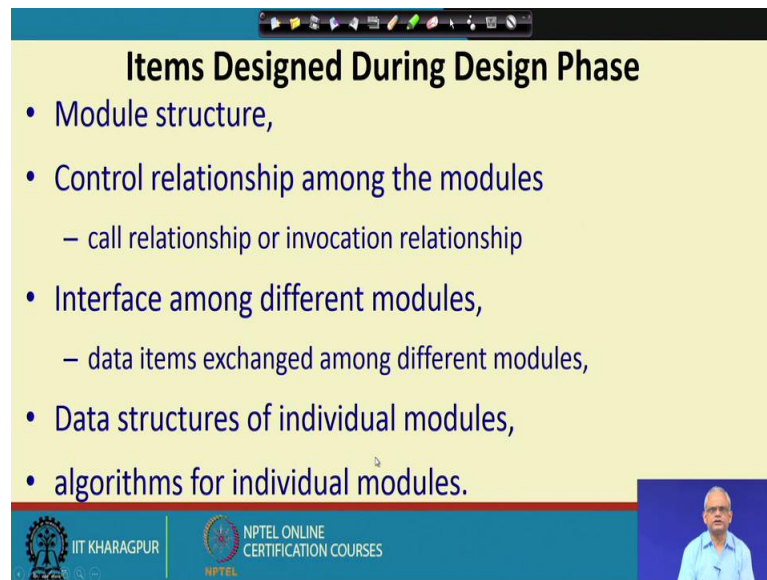
(Refer Slide Time: 07:00)



During the design phase, we have the SRS documents as the input and then do some activities during the design and then, we come up with the design document at the end of the design phase. The design document is taken up for coding purpose and our design document is good if the programmers can take it and can start writing the code easily. A graphic representation of the design phase is shown in the slide.

The circle in the slide represents design phase and the input of the design phase is the SRS document. We perform some design activities and then, at the end of the phase, we have the design document. Now let see at the end of the phase what are the documents we should have with us? What we should prepare during the design phase and then will address the question how to prepare those?

(Refer Slide Time: 08:43)



**Items Designed During Design Phase**

- Module structure,
- Control relationship among the modules
  - call relationship or invocation relationship
- Interface among different modules,
  - data items exchanged among different modules,
- Data structures of individual modules,
- algorithms for individual modules.

The slide features a blue header with a navigation bar. The main content area is yellow. At the bottom, there is a blue footer with logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES. A small video inset in the bottom right corner shows a man in a light blue shirt speaking.

Now, we will see items designed during design phase. Typically, we have the module structure in a procedural design or the class structure in an object-oriental design. In a procedural design, we have the control relationship among the modules and similarly in an object-oriental design, we have the invocation sequences among different classes. Call relationship or invocation relationship between the modules are basically control relationship among modules. Then we have the interface that is what data exactly is a given as a parameter when there is a function call or a method call that we call as the interface among the modules. So, this is basically the data items that are exchanged between the modules. And also, we need to design the data structure of the individual modules and also the algorithms that would be used. So, these are all the different items that must be designed during the design phase.

The call relationship among the module in a object oriented way we will see it little later that this is a class structure, the invocation relationship among the classes. And then, the interface among the modules that is what parameters, they pass or the data items that are exchanged among the different modules. The data structures of the individual modules and then the algorithms for the individual modules.

(Refer Slide Time: 10:53)

The slide is titled "Module" in a large, bold, black font. Below the title, there is a bulleted list: "• A module consists of:" followed by two sub-points: "– several functions" and "– associated data structures." To the right of the list is a diagram of a module. It is a vertical rectangle divided into two main sections. The top section is green and labeled "Data" in yellow. It contains a list: "D1 ..", "D2 ..", and "D3 ..". The bottom section is blue and labeled "Functions" in yellow. It contains a list: "F1 ..", "F2 ..", "F3 ..", "F4 ..", and "F5 ..". The word "Module" is written in large, bold, black font across the bottom of the blue section. At the bottom of the slide, there is a blue banner with the IIT Kharagpur logo on the left and the text "NPTEL ONLINE CERTIFICATION COURSES" on the right. A small video inset of a man in a blue shirt is in the bottom right corner.

## Module

- A module consists of:
  - several functions
  - associated data structures.

D1 ..	Data
D2 ..	
D3 ..	
F1 ..	Functions
F2 ..	
F3 ..	
F4 ..	
F5 ..	

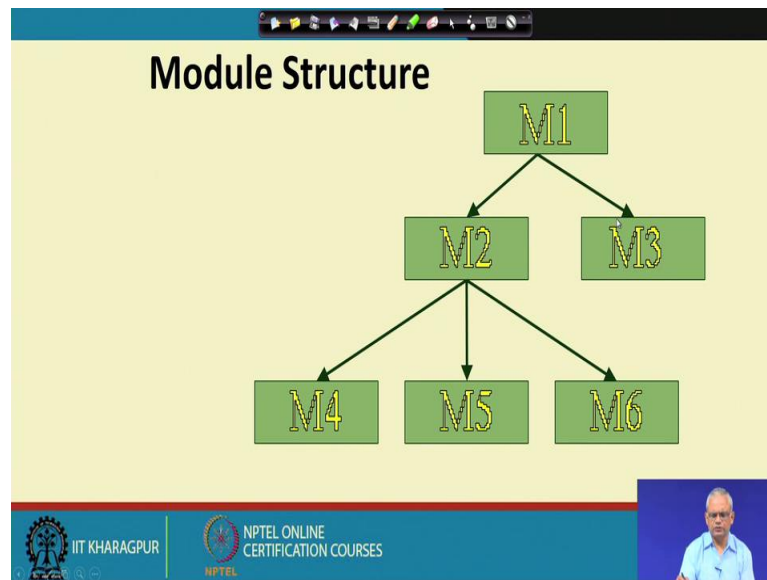
Module

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, far we have been referring modules; but what exactly is a module? In a procedural implementation, a module is basically several functions and then there are associated data structures which are referred by the functions. In the context of object-oriented designs, a module can be classed or a package. So, in a procedural design module consists of several functions and there are some global data may be arrays, link lists or the other structure and these different functions can access those global data.

So, a module consists of certain data structures and also the functions and in a procedural implementation, a module is an independently compilable unit. We can create the object code for a module, we can compile it; then after executing code, we can link the object codes for various modules.

(Refer Slide Time: 12:16)



In the slide, call relationships among the different modules are shown. We will see that it is a good design if our module structure looks like this. Procedural implementation looks like a tree. As we proceed, we will understand the reasons why the module structure or the call relationship among the modules should look like a tree.

(Refer Slide Time: 12:53)

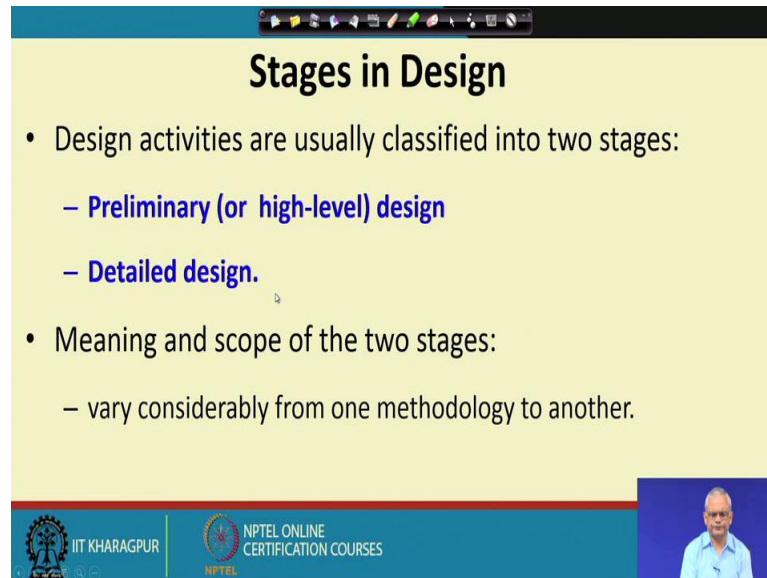
The slide is titled "Iterative Nature of Design". It contains a bulleted list of points about good software designs. The slide includes logos for IIT Kharagpur and NPTEL Online Certification Courses at the bottom.

- Good software designs:
  - Seldom arrived through a single step procedure:
  - But through a series of steps and iterations.

Another thing, we must keep in mind is that when we try to design all the items that we discussed, the modules structure, the call relationships, the parameters, data structures,

the individual modules, algorithms to be used and so on, it's never come as a sequence. Design is a very intellectually stimulating work and need to do several iterations.

(Refer Slide Time: 13:30)



**Stages in Design**

- Design activities are usually classified into two stages:
  - Preliminary (or high-level) design
  - Detailed design.
- Meaning and scope of the two stages:
  - vary considerably from one methodology to another.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, as we look at the procedural and object-oriented design, we will see that we will have to iterate over several steps. Look at evaluate designs alternatives and then we will come up with a good design. Now let's look at what are the stages in the design? There are two main stages in the design phase: one is called as the high-level or the preliminary design and the other is called as the detailed design.

The high-level design is also called as architectural design and then we have the detailed design. So, first we have preliminary or high-level design or an architectural design and then, we have the detailed design. As we will see, that there is a significant difference in the steps, in the procedural and object-oriented design. And the meaning and scope of these high-level and detailed design vary quite widely between procedural and object-oriented design.

Even in the procedural design, there are several methodologies. We look at one of the methodologies; but there exist dozens of methodologies. Even among the methodologies, there is quite a variation in what should be done the architectural design and what exactly to be done in the detailed design. We look those issues little later.

(Refer Slide Time: 15:36)

## High-level design

- Identify:
  - modules
  - control relationships among modules
  - interfaces among modules.

```
graph TD; M1[M1] -- d1 --> M2[M2]; M1 -- d2 --> M3[M3]; M2 -- d3 --> M4[M4]; M2 -- i1 --> M5[M5]; M2 -- d4 --> M6[M6];
```

The diagram illustrates a module hierarchy. Module M1 is the root, with two children: M2 and M3. M2 has three children: M4, M5, and M6. The arrows representing the relationships are labeled: d1 (M1 to M2), d2 (M1 to M3), d3 (M2 to M4), i1 (M2 to M5), and d4 (M2 to M6).

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, let see what is expected in the high-level design? In the high-level design, after we complete this task, we should have come up with the module structure. What are the modules in our implementation and their call relationships and the interfaces between the modules? But then, how do we come up with the modules; how do we determine their call structure and what are the parameters to be exchanged that we will discuss subsequently.

(Refer Slide Time: 16:31)

## High-level design

- The outcome of high-level design:
  - program structure, also called software architecture.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES



The high-level design is also called as a software architecture. One of the popular notations for representing the high-level design is a structure chart.

(Refer Slide Time: 16:41)

### High-level Design

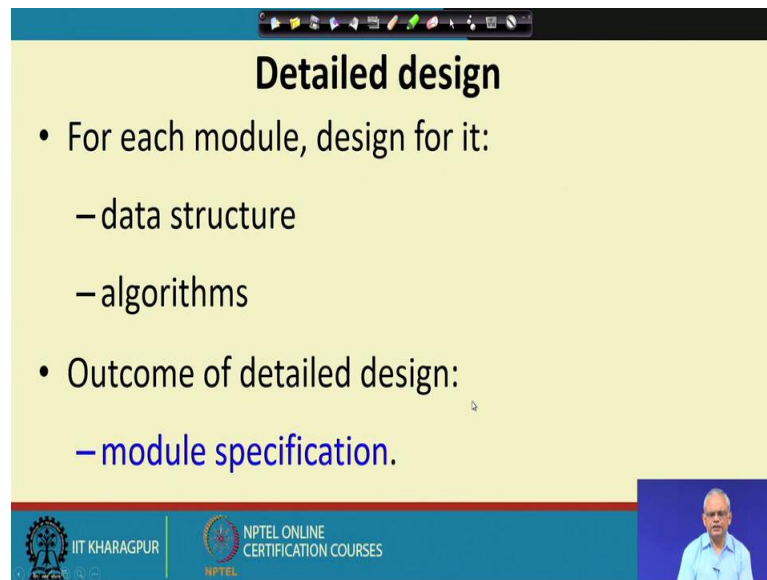
- Several notations are available to represent high-level design:
  - Usually a tree-like diagram called **structure chart** is used.
  - Other notations:
    - Jackson diagram or Warnier-Orr diagram can also be used.

```
graph TD; M1[M1] -- d1 --> M2[M2]; M1 -- d2 --> M3[M3]; M2 -- d3 --> M4[M4]; M2 -- d1 --> M5[M5]; M2 -- d4 --> M6[M6];
```

The slide also features logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES at the bottom, along with a small video inset of a speaker.

Structure chart is very straight forward. It looks like a tree structure with modules as a rectangle. We look at the structure chart representation of the high-level design in the slide and how to come up with the structure chart representation that we will address next. The structure chart is not the only representation that is possible for the high-level design. There are many other notations: for example, a Jackson diagram, Warnier-Orr diagram and so on.

(Refer Slide Time: 17:39)



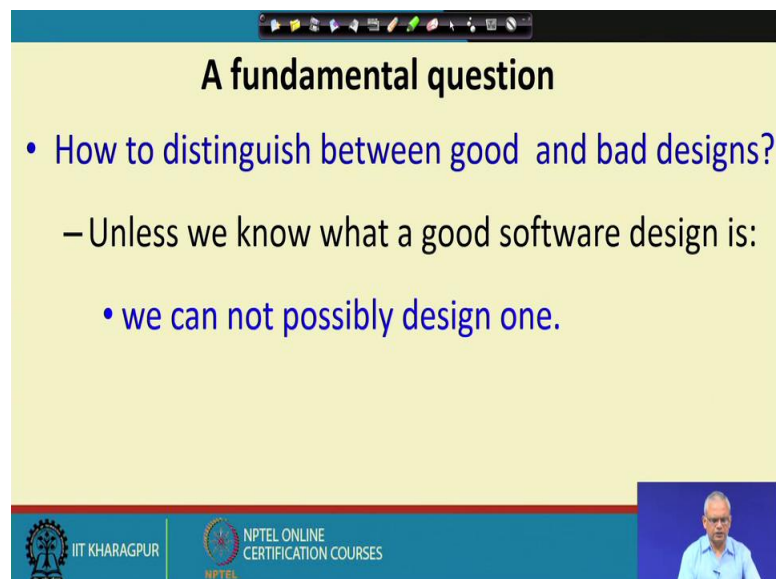
**Detailed design**

- For each module, design for it:
  - data structure
  - algorithms
- Outcome of detailed design:
  - module specification.

The slide features a yellow background with a blue header and footer. The footer contains the IIT Kharagpur and NPTEL logos. A small video inset in the bottom right corner shows a man in a light blue shirt speaking.

Once the high-level design is complete or the architectural design is complete, the detailed design is carried out. In detailed design, we look at what are the modules that has been identified during the high-level design and then for each of the module need to design the data structure and also the algorithms to be used.

(Refer Slide Time: 18:00)



**A fundamental question**

- How to distinguish between good and bad designs?
  - Unless we know what a good software design is:
    - we can not possibly design one.

The slide features a yellow background with a blue header and footer. The footer contains the IIT Kharagpur and NPTEL logos. A small video inset in the bottom right corner shows a man in a light blue shirt speaking.

The outcome of the detailed design is called as the module specification. This is a module specification there are various notations for module specification. It can be a

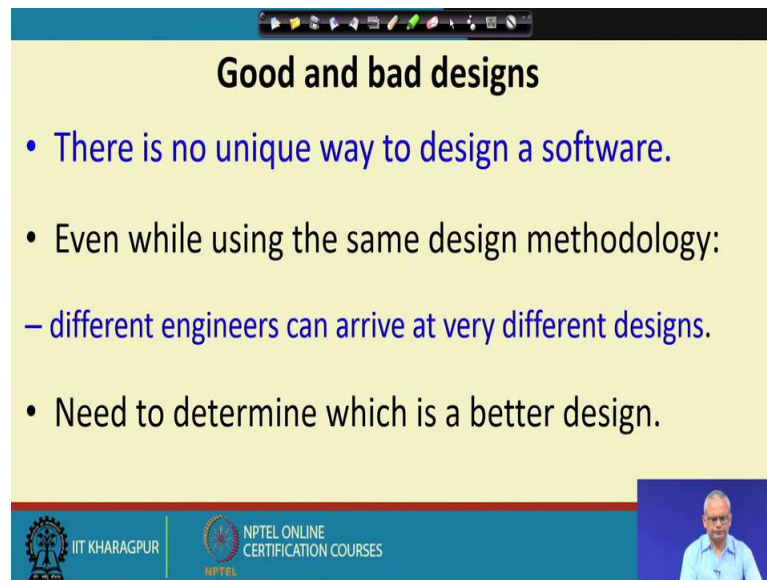
simple text-based notation in formal notation, it can be slightly semi formal notations where represent the data structures and the algorithms.

So far, we just looked at some simple concepts of design phase. We saw that during design, we need to have the high-level or the architectural design that is the first step and once the architectural design is complete, we have the module structure call invocation, call structure. The invocation among the different modules is the data exchange between different modules. And we represent that in a tree like diagram that is a structure chart and after that take up the detailed design where we design the data structure of the individual modules and also the algorithms.

But then, let's try to understand a very fundamental question that let say you have come up with a design, a tree like diagram you have represented, come up with the module specification; but then how do you know that whether your design is good or bad? or what should be the target of any designed methodology to come up with an ideal design? Because unless we know that what we mean by a good software design, even if we have given to learn the methodologies of design and so on, we will not come up with a good design.

Because ourselves before design, we have to be clear about what is meant by a good design; unless we understand that issue that what is a good design, we cannot come up with a good design. Even if we use the most sophisticated tools and design methodologies.

(Refer Slide Time: 20:32)



**Good and bad designs**

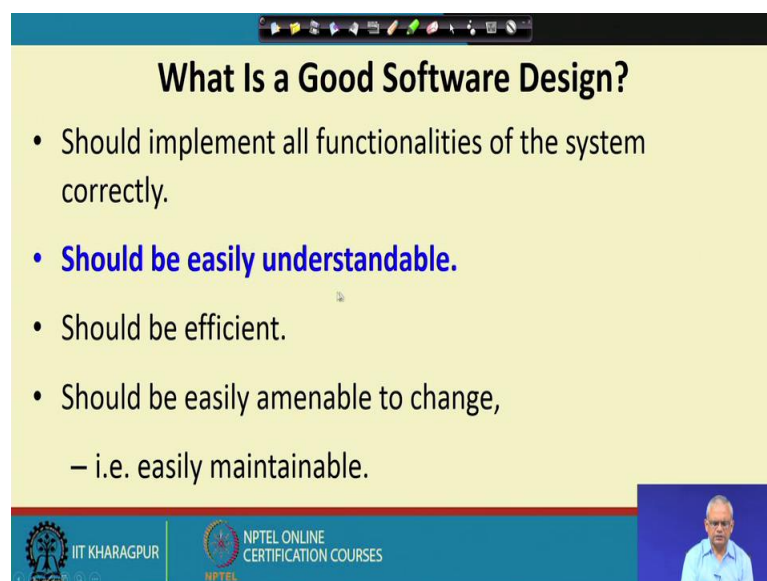
- There is no unique way to design a software.
- Even while using the same design methodology:
  - different engineers can arrive at very different designs.
- Need to determine which is a better design.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

The slide features a yellow background with a blue header and footer. A video inset in the bottom right corner shows a man in a light blue shirt speaking. The footer includes the IIT Khargapur logo and the NPTEL Online Certification Courses logo.

Another reason why we must know about good designs is that we must be able to distinguish between the good and bad designs or in other words, we should be evaluate between design alternatives. During the various design steps, we will see that there are alternate solutions are possible. We should be able to take the better ones because it's not a unique way to design a software. We will see that always evaluating between alternatives and for evaluating between alternatives, we need to distinguish between which is better design and which is a inferior design.

(Refer Slide Time: 21:30)



**What Is a Good Software Design?**

- Should implement all functionalities of the system correctly.
- **Should be easily understandable.**
- Should be efficient.
- Should be easily amenable to change,
  - i.e. easily maintainable.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

The slide features a yellow background with a blue header and footer. A video inset in the bottom right corner shows a man in a light blue shirt speaking. The footer includes the IIT Khargapur logo and the NPTEL Online Certification Courses logo.

Another reason is that it may be possible that different developers, by using the same methodology they can come with very different designs in the end. We should be able to tell how good is the design or we can select the better design if we have different designers come up with different designs.

So, let's now address this important question that what is a good software design?

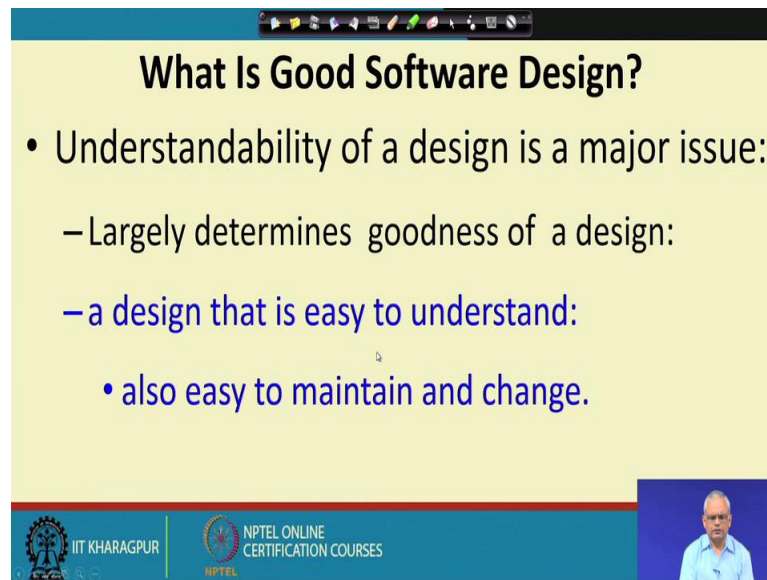
We will answer this very conceptually. The first characteristic of a good software design is that it should have addressed all the requirements that were there in the requirements documents. In other words, it should implement all functionalities of the system that are specified in the SRS document.

So, the functionalities et cetera should have been correctly understood and our designs should be as per the requirement. First characteristic is that design needs to be correct, unless design is correct, it's no good. The second characteristic of a good design is that it should be easily understandable; it should not look like a messy structure and somebody spends days, months or years trying to understand the design.

It should be an elegant design and somebody looks at the design should be able to easily understand the design. Understandability is a major characteristic of a good software design. Of course, it should result in an efficient solution and another characteristic may be that it should be able to change the design easily.

As we know, during development a lot of changes happen and even after the development the software continues to change and therefore, our design should be such that we would be able to change any part of the software and that helps in maintaining the software. But we have highlighted one aspect here is that the design should be easily understandable and it turns out that this is one of the most important characteristics of a good software design that it should be easily understandable.

(Refer Slide Time: 24:51)



**What Is Good Software Design?**

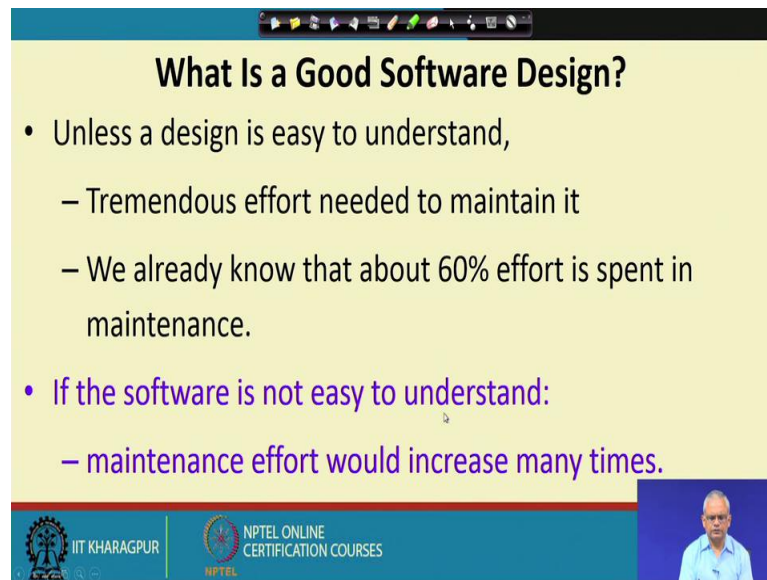
- Understandability of a design is a major issue:
  - Largely determines goodness of a design:
  - a design that is easy to understand:
    - also easy to maintain and change.

The slide is a presentation slide from NPTEL. It has a yellow background with a blue header and footer. The title is 'What Is Good Software Design?'. The main content is a bulleted list. The first bullet point is 'Understandability of a design is a major issue:'. It has two sub-bullets: 'Largely determines goodness of a design:' and 'a design that is easy to understand:'. The second sub-bullet has a further sub-bullet: 'also easy to maintain and change.'. In the bottom right corner, there is a small video inset showing a man in a light blue shirt. The bottom left corner has the IIT Kharagpur logo and the NPTEL logo. The bottom center has the text 'NPTEL ONLINE CERTIFICATION COURSES'.

The main reason why we consider understandability of a design is a major issue is that unless somebody able to understand the design, he or she cannot write code for that and also becomes very difficult to maintain the design. If the maintainers cannot understand the design then they won't know where to change, what to change et cetera. We also know that maintenance is a major concern because majority of the effort and cost is incurred the maintenance work compared to the development work or that we can say that development work is a fraction of the maintenance work and therefore, while writing any software, whether it is maintainable is a major concern and to facilitate maintenance, we have to develop design which is simple to understand.

Understandability of the design is a major issue, require significant attention by the designers and also every design methodology helps to come up with good design which is easily understandable.

(Refer Slide Time: 26:36)



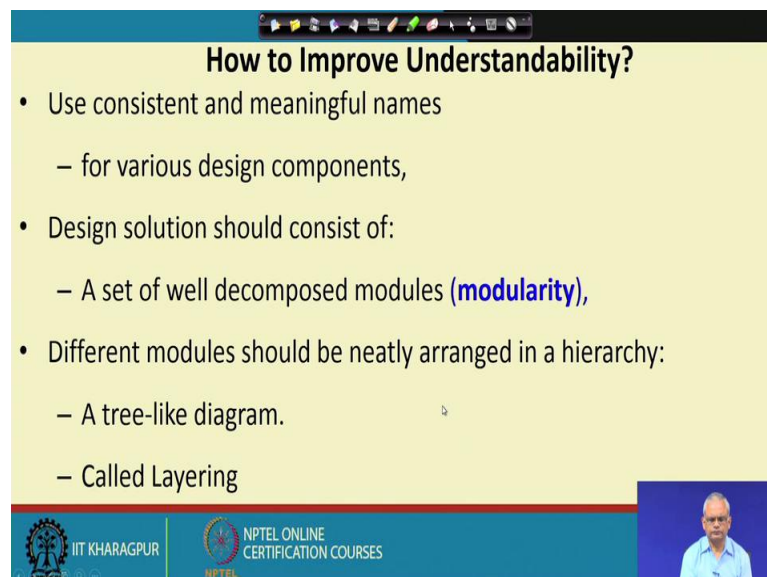
### What Is a Good Software Design?

- Unless a design is easy to understand,
  - Tremendous effort needed to maintain it
  - We already know that about 60% effort is spent in maintenance.
- If the software is not easy to understand:
  - maintenance effort would increase many times.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Majority of the effort spent on maintenance and therefore, unless design is understandable, maintenance effort will become extremely expensive.

(Refer Slide Time: 26:56)



### How to Improve Understandability?

- Use consistent and meaningful names
  - for various design components,
- Design solution should consist of:
  - A set of well decomposed modules (**modularity**),
- Different modules should be neatly arranged in a hierarchy:
  - A tree-like diagram.
  - Called Layering

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

But the next question that we will answer is that how to improve understandability? Or how do we come up with a design which has good understandability?

First let us agree that one of the most desirable characteristics of any design solution is that it should be very easy to understand. And then, we will have to address the question that how do we come up with a design that has good understandability?

This is an issue which has some details that we must understand and in the next lecture, we will address this very basic issue that how does one come up with a design that will have good understandability.

We will stop here and continue in the next lecture.