

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

Lecture – 20
Modular Design

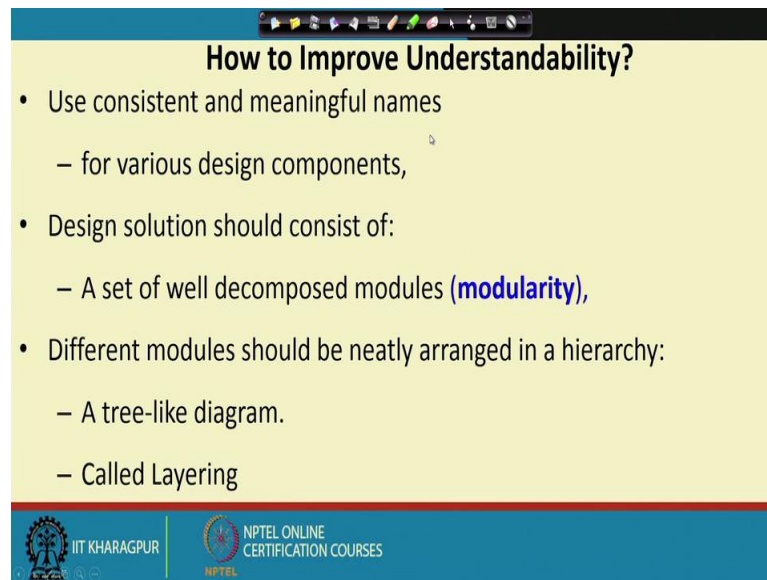
Welcome to this lecture. In the last lecture, we had looked at some very basic issues in software design. We said that it's important to distinguish a good design from a bad design. And, then we are trying to characterize, what is a good design? And, then we said that there are many factors that determine, whether design is good or bad. But correct implementation of the functionality that possibly the important requirement of a good design, because unless it is a correct design, it's not a good design.

But there are several design alternatives which can be factor for a good design. We can come up with different alternate designs, which are all correct, but then how do we decide which is a better design? And, then we had said that understandability of a design is a major issue. We can rate a design solution to be good or bad based on it is understandability.

But there are two questions that arise now. That how do we know, that, which design is more understandable? It's a debatable question, because somebody may say that this design is more understandable, another person will say that no that's not a very understandable design. So, we should be able to tell more formally what do you mean by an understandable design?

And, also, we have to address this question that how do we really come up with an understandable design? So, let's proceed from that. First let's look at the characteristics of a design, that enhance it is understandability. So, the first characteristic is the use of consistent and meaningful names in the design.

(Refer Slide Time: 02:44)



How to Improve Understandability?

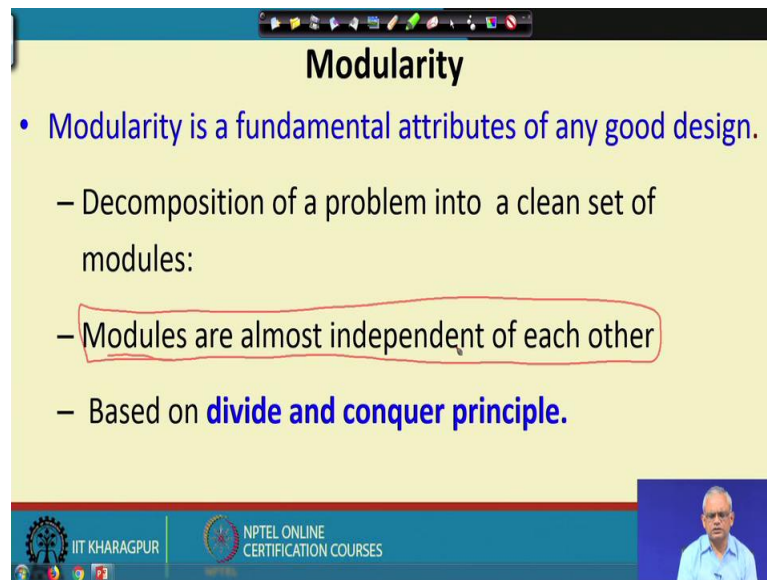
- Use consistent and meaningful names
 - for various design components,
- Design solution should consist of:
 - A set of well decomposed modules (**modularity**),
- Different modules should be neatly arranged in a hierarchy:
 - A tree-like diagram.
 - Called Layering

The slide features a blue header with a navigation bar, a yellow main content area, and a blue footer. The footer contains the IIT Kharagpur logo and the text 'NPTEL ONLINE CERTIFICATION COURSES'.

Because, unless the names in the design are meaningful in the problem context, anybody trying to understand the design will get confused. All design components should have names, that corresponds to the problem domain as far as possible and they should be meaningful so anybody can understand.

The second characteristic of a design which is understandable is that it should have been decomposed well into modules. This we call it as the modularity of the design. We need to look at this question that what do we mean by modularity? This is an important characteristic that the design should be modular. And, the modules should have been such that they have been well designed. Just arbitrary set of modules is not a good design; the module should have been well decomposed. And, also the call relationship should look like a tree diagram, this is also called as layering. Because, each layer of the tree is an implementation of the previous layer and a specification of the lower layer. This is a desirable characteristic we will see later that it is not only that we should have a well decomposed set of modules, but also their call relation should be representable in the form of a tree like diagram.

(Refer Slide Time: 05:05)



Modularity

- Modularity is a fundamental attributes of any good design.
 - Decomposition of a problem into a clean set of modules:
 - Modules are almost independent of each other
 - Based on **divide and conquer principle**.

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

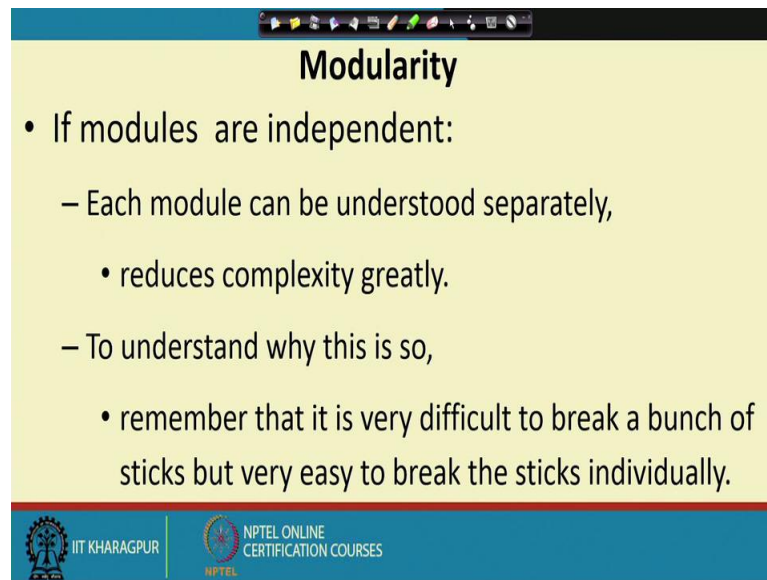
Now, let's try to understand, what we mean by modularity or well decomposed set of modules? We say that a design is modular, if the modules are almost independent of each other. A set of modules are almost independent of each other, if they either do not call any other module or they call very less. If, a module calls another module, and it requires many parts to be completed by other modules then it becomes a dependent module.

In a well decomposed set of modules, the module should be as independent as possible. It's not possible to have a completely independent set of modules that means no module interacts with other, but the interaction among the module should be kept to a minimum. To understand why it is so, we will argue that this is actually the divide and conquer principle.

A modular design uses the divide and conquer principle to be able to understand the design. We just take up each module and then we understand. And, if after we have understood all the modules, we have understood the design.

But, if the modules have complex relations that is each module calls several other modules then while understanding a single module, we need to also see what it gets done by the other modules and so on. So, as long as the modules don't interact too much, we can understand them well based on the divide and conquer principle.

(Refer Slide Time: 07:34)



Modularity

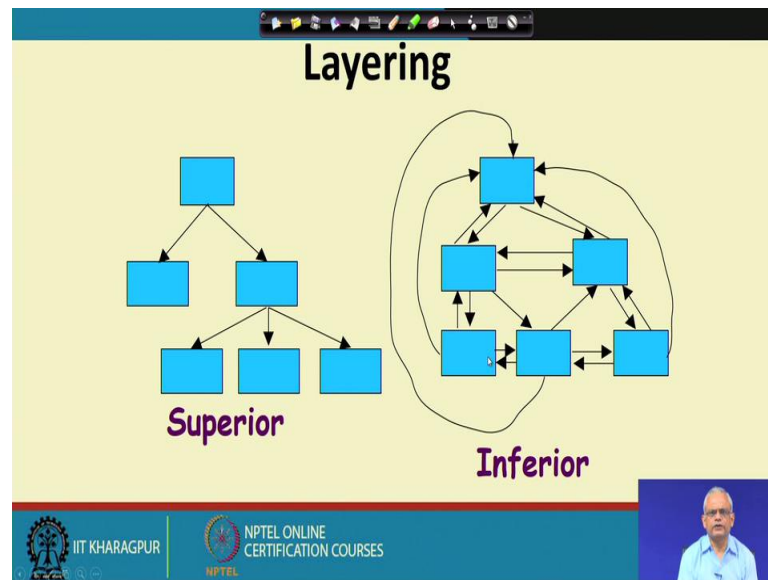
- If modules are independent:
 - Each module can be understood separately,
 - reduces complexity greatly.
 - To understand why this is so,
 - remember that it is very difficult to break a bunch of sticks but very easy to break the sticks individually.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, if the modules are independent, we can look up the modules one by one and understand this and this is based on the design and divide and conquer principle.

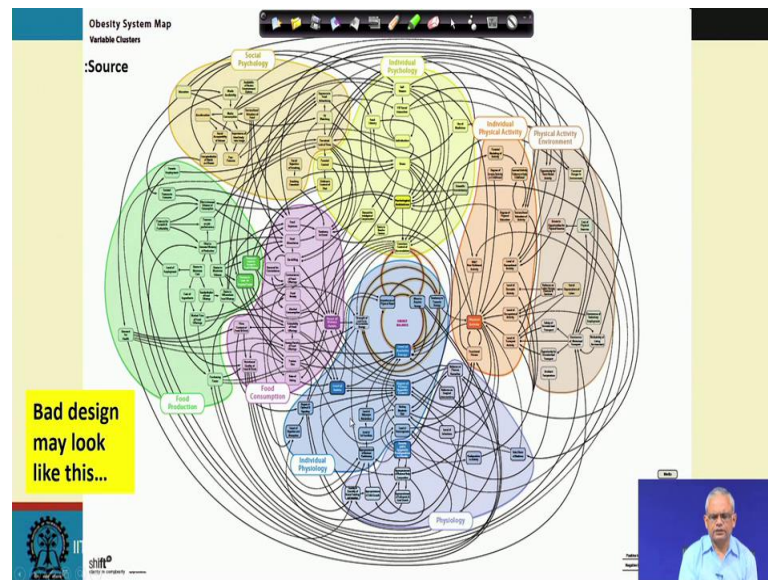
If a bunch of sticks are tied together and we tried to break it, then it becomes very difficult, but if we take one stick at a time and break it, then becomes very easy. It's thus the similar thing here is that if we tried to understand module one by one, we can easily understand, but trying to understand all modules which are interconnected with each other, then if we start with one module we see that we need to understand another module and the third module and so on.

(Refer Slide Time: 08:44)



Another requirement of a good design is layering. A tree like call relationship among modules is a superior, whereas an arbitrary call relation is an inferior design. We will see why the layering is a good idea: one is bug localization. In superior layering, if there is a bug in any intermediate node, it does not affect the other modules except its child module. Whereas in inferior layering, any bug in any node can have effect on any other module. And, when we see there is a failure, we don't know what is the module, that is having the bug? But in superior layering, if we observe a bug in any leaf node, we know that either the bug is in that node or in the previous parent node or in any upper node. In superior layering, we do not have to look on all nodes.

(Refer Slide Time: 10:05)



A bad design can come up with a very complex set of module interactions. We can see in the slide that a complex module interaction has shown. The slide diagram, is not really a set of modules and interactions, it is a diagram for some other purpose. But here we are showing it for illustration, that it can look extremely odd and here for understand one module, you might have to understand all the modules associated with the module we want to understand. So, for understanding one module we need to trace all the modules and it becomes very difficult to understand. So we can say that to understand a module we need a nice tree like diagram.

(Refer Slide Time: 10:55)

Modularity

- In technical terms, modules should display:
 - high cohesion
 - low coupling.
- We next discuss:
 - cohesion and coupling.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, far we said that modularity is a set of independent modules and then we said that independent modules are actually not practical. There will be some interactions between modules, but can we give a more meaningful characterization of modularity. We had intuitive understanding of what is modular design? that modules are almost independent of each other, don't depend too much represented in a tree like diagram and so on.

Let's try to give a better characteristic or more precise characteristic of what is a modular design? And, we will do that in terms of two concepts: one is called as cohesion and the other is called as coupling. A modular design should have high cohesion and low coupling.

And, if we understand these terms cohesion and coupling and we can measure them, then for a given design solution, we can check whether it has high cohesion and low coupling.

(Refer Slide Time: 12:25)

Modularity

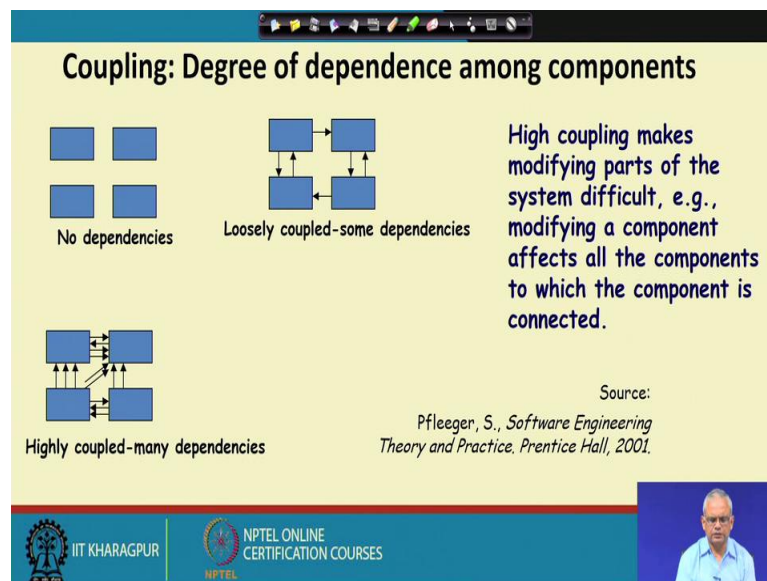
- Arrangement of modules in a hierarchy ensures:
 - Low fan-out
 - Abstraction

The diagram illustrates a hierarchical structure on the left, where a top module connects to two lower modules, one of which further connects to two more. On the right, a circular cluster of modules is shown with multiple interconnecting arrows, representing high coupling and low abstraction. The slide footer includes the IIT Kharagpur logo and NPTEL Online Certification Courses text, with a small video feed of a speaker in the bottom right corner.

So, let's try to understand how to measure, define cohesion and coupling? The module should be layered. A good layering means a tree like diagram. But then let me just mention briefly, that in a tree like diagram, if, we want to let say understand one module, then we see that it calls only the lower modules and for that maybe we will have to look at lower modules. If, we want to understand any intermediate module then we do not even have to look at upper module. Because, we know that intermediate will call only its lower layer modules. So in tree like structure, we can understand each module through top to down hierarchy.

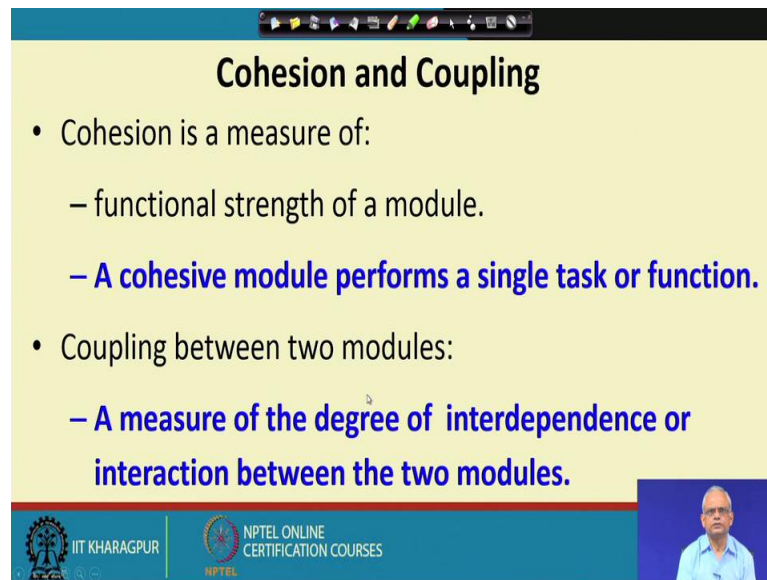
But, if it is a module structure like 2nd drawn diagram in slide, then to able to understand one we need to just go across all other modules and so on. So, it becomes very difficult to understand each module. So, a tree like hierarchy, we have low fan-out that is it depends on a smaller number of modules. And, also there is an abstraction. Abstraction in the sense that in tree like structure, for implementation, we can start from the leaf layer modules and then look at more concrete module and so on.

(Refer Slide Time: 14:54)





Above slide, is an illustration of how bad it can become if the modules are having lot of dependencies. In slide, first one is showing no dependencies, second one has very less number of dependencies, but look at the third one, there are very high dependencies across modules and it becomes very difficult to understand. So, maintain this kind of design like 3rd one is very difficult. So, we would rather go for 1st one if not, we will go for 2nd one.

(Refer Slide Time: 15:32)



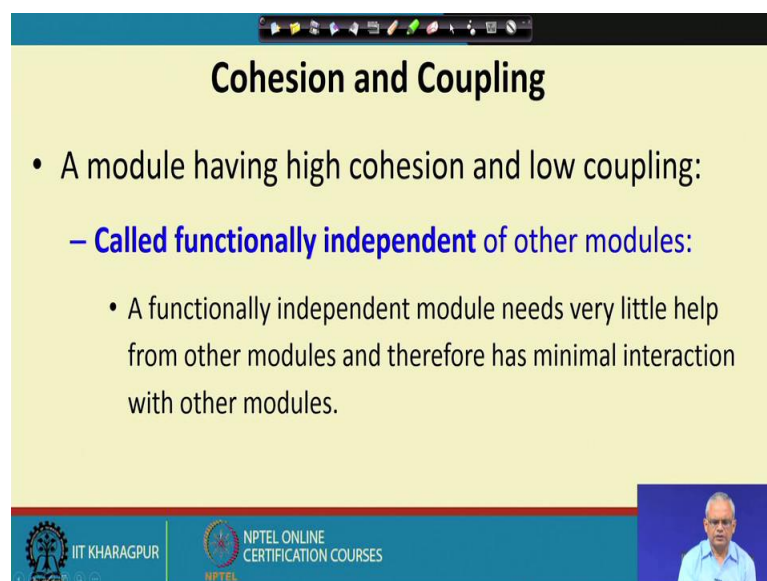
Cohesion and Coupling

- Cohesion is a measure of:
 - functional strength of a module.
 - **A cohesive module performs a single task or function.**
- Coupling between two modules:
 - **A measure of the degree of interdependence or interaction between the two modules.**

 IIT KHARAGPUR  NPTEL ONLINE CERTIFICATION COURSES



Now, let see the concepts of cohesion and coupling. If, we can determine the cohesion and coupling of a given design solution, we can tell that whether that's a good design or bad design. We say that a module is cohesive, if it performs a single task or function. If, it tries to do too many things then it's not cohesive. On the other hand, the coupling is defined between two modules. Two modules are coupled if they dependent on each other. Dependent on each other means they exchange some data items, call each other etc. So, this is two important concepts. Cohesion is defined it for a single module.

(Refer Slide Time: 16:43)



Cohesion and Coupling

- A module having high cohesion and low coupling:
 - **Called functionally independent of other modules:**
 - A functionally independent module needs very little help from other modules and therefore has minimal interaction with other modules.

 IIT KHARAGPUR  NPTEL ONLINE CERTIFICATION COURSES

And then we say that a single module is cohesive, if it performs a single task, if it does many things then that is not really cohesive. Coupling is that how much dependent one module is on another module. If, they exchange too many data items call each other frequently exchanging data items and so on then, they are highly coupled. But if they don't call each other, there is no data exchange between two modules, we say that these two modules are independent. So, if there is a high cohesion and low coupling then we say that this is a good design.

Now, let's look at how do we determine the cohesion and coupling? Before, that if a design solution we find that it has high cohesion and low coupling, then we say that the design is a functionally independent set of modules. So, in the design, each module is doing some function independently that means it's not dependent on others. And the term that we use for this is functionally independent.

So, the goal of any good design technique is to come up with a functionally independent set of modules.

(Refer Slide Time: 18:32)

Advantages of Functional Independence

- Better understandability
- Complexity of design is reduced,
- Different modules easily understood in isolation:
 - Modules are independent

The slide includes two diagrams. The first diagram, labeled 'No dependencies', shows four blue rectangular modules arranged in a 2x2 grid with no connecting lines between them. The second diagram, labeled 'Highly coupled-many dependencies', shows four blue rectangular modules arranged in a 2x2 grid with multiple arrows between them, indicating frequent data exchange. The slide footer features the IIT Kharagpur logo and the text 'NPTEL ONLINE CERTIFICATION COURSES'.

But we should be clear why we want a functionally independent set of modules? As, we said that functionally independent results in better understandability compared to a highly dependent set of modulus.

If, we have a functionally independent set of modules the complexity in the design is reduced and we can take up one module at a time quickly to understand. Similarly, if there is a bug in the 1st diagram of the below slide, we can easily know that which module contains the bug, otherwise we might see that whether it has come from, may be some module to which it has called. But if there is a bug while executing the 2nd design module of the below slide, we can't easily identify the module which contain the bug.

(Refer Slide Time: 19:43)

Why Functional Independence is Advantageous?

- Functional independence reduces error propagation.
 - degree of interaction between modules is low.
 - an error existing in one module does not directly affect other modules.
- Reuse of modules is possible.

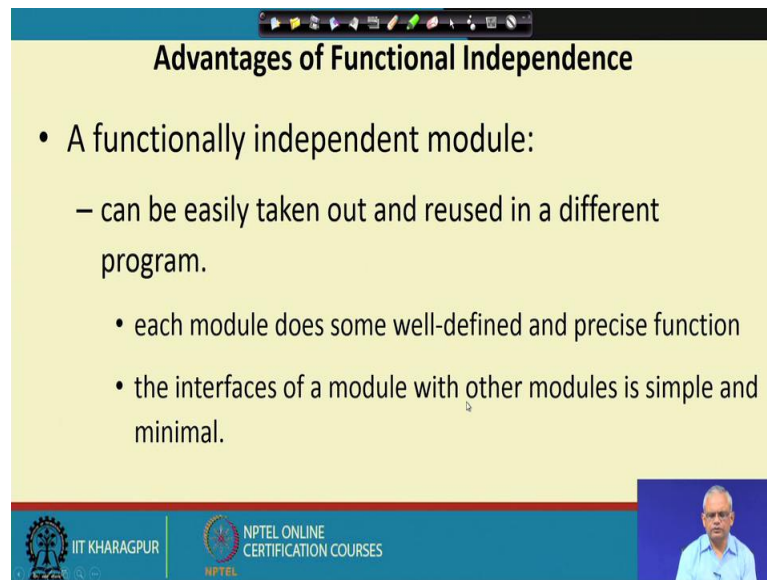
No dependencies

The slide features a presentation interface with a toolbar at the top. The bottom of the slide contains logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, along with a small video feed of a presenter.

So, if the modules are functionally independent, debugging becomes easy and also understandability become easy and not only that, we can re-use module.

Let say, in the 1st diagram of above slide, we want to use any module in another project, we can just take the module and use it there. But here observe it if we want to use any module of 2nd diagram in another project. We will see that the needs help from another module and another module in turn needs help from another and so on. So, they are all coupled. We cannot take just one module and use it we have to take across all these modules that entire project we have to take and that becomes infeasible. So, if we have a functionally independent set of modules, any module that we need in another project, we can easily re use that.

(Refer Slide Time: 20:56)



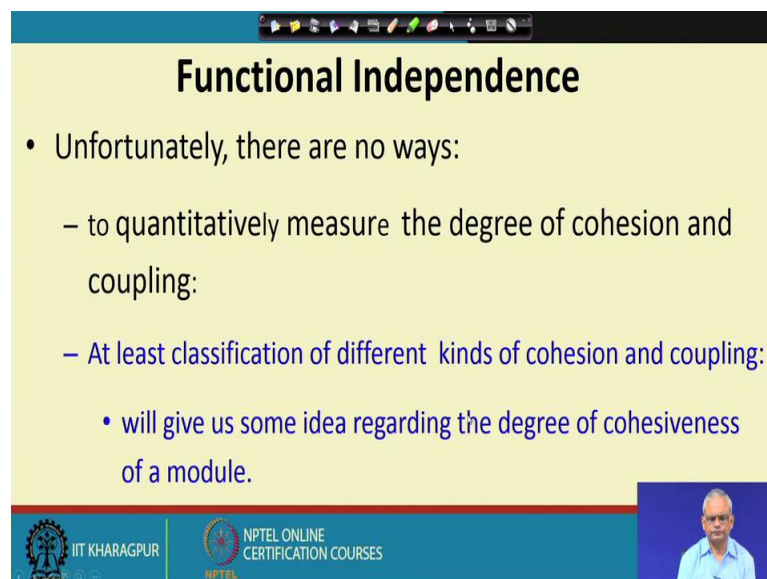
Advantages of Functional Independence

- A functionally independent module:
 - can be easily taken out and reused in a different program.
 - each module does some well-defined and precise function
 - the interfaces of a module with other modules is simple and minimal.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Each module in a functionally independent design, has well defined precise functions. And, it does not interact with other modules and it interfaces with other modules simple and minimal. And therefore, becomes easy to re use modules.

(Refer Slide Time: 21:20)



Functional Independence

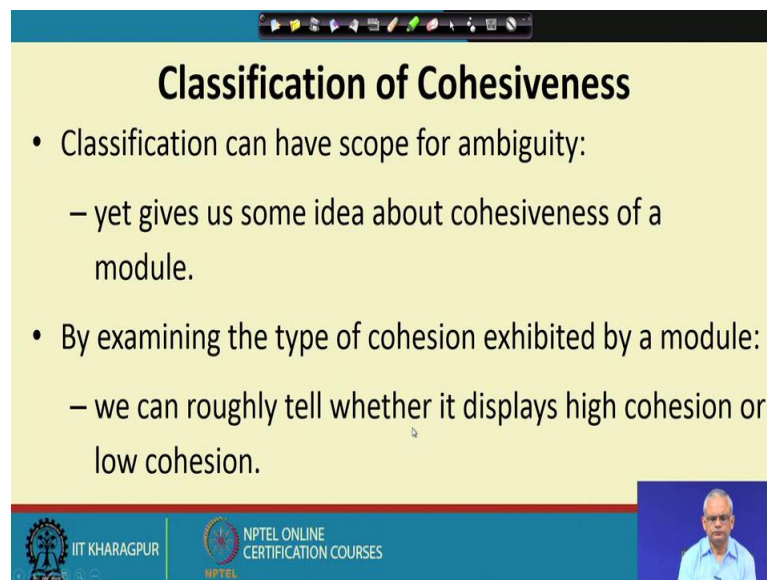
- Unfortunately, there are no ways:
 - to quantitatively measure the degree of cohesion and coupling:
 - At least classification of different kinds of cohesion and coupling:
 - will give us some idea regarding the degree of cohesiveness of a module.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

But, how do we measure functional independence? Even though we know that a functional independent set of module has high cohesion and low coupling. Can, we look at a design and say that whether there is a high cohesion and low coupling?

Let's try to explore this further; we will classify the different types of cohesion and coupling. So, that given a design we will try to see, which class of cohesion and coupling it has and based on that we will say that, whether it is a cohesive, highly cohesive and low coupling et cetera.

(Refer Slide Time: 22:09)



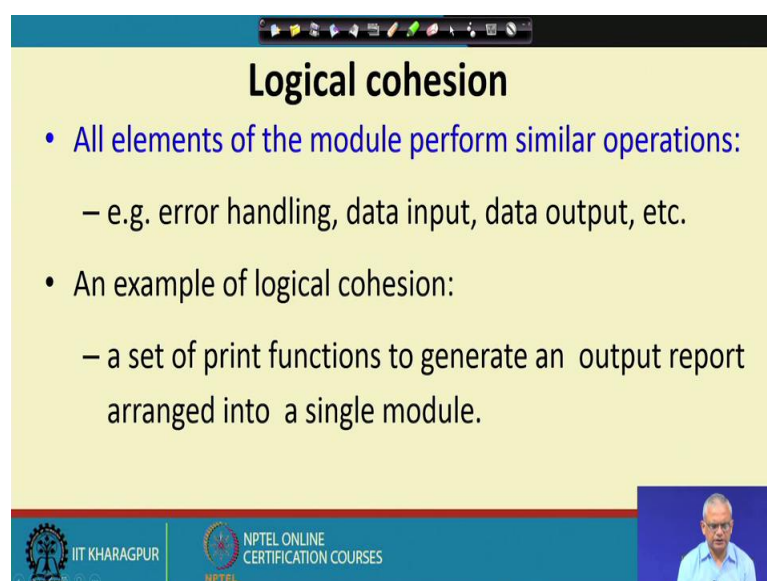
Classification of Cohesiveness

- Classification can have scope for ambiguity:
 - yet gives us some idea about cohesiveness of a module.
- By examining the type of cohesion exhibited by a module:
 - we can roughly tell whether it displays high cohesion or low cohesion.

The slide features a blue header with navigation icons, a yellow main content area, and a blue footer. The footer contains the IIT Kharagpur logo, the text 'IIT KHARAGPUR', the NPTEL logo, and the text 'NPTEL ONLINE CERTIFICATION COURSES'. A small video inset in the bottom right corner shows a man in a light blue shirt speaking.

First let's look at the cohesiveness and let's try to classify what are the type of cohesion? So, that if we look at a design, we will see that what set of cohesion it has.

(Refer Slide Time: 22:26)



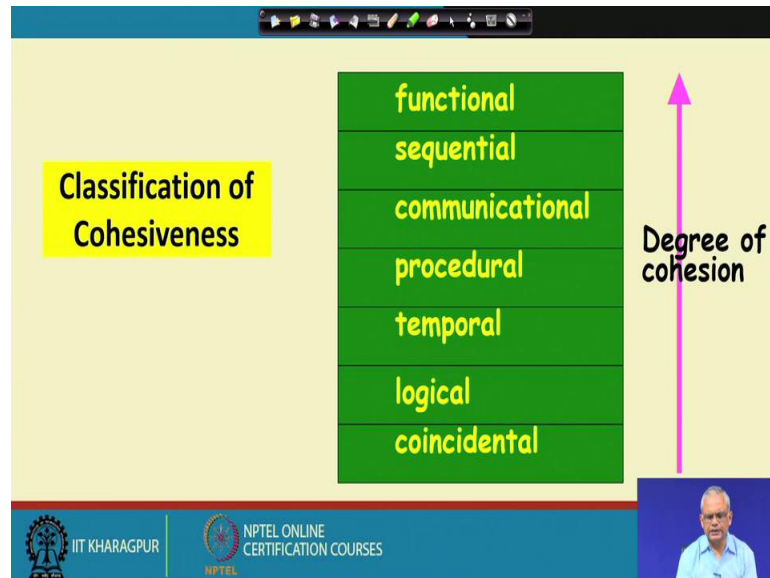
Logical cohesion

- All elements of the module perform similar operations:
 - e.g. error handling, data input, data output, etc.
- An example of logical cohesion:
 - a set of print functions to generate an output report arranged into a single module.

The slide features a blue header with navigation icons, a yellow main content area, and a blue footer. The footer contains the IIT Kharagpur logo, the text 'IIT KHARAGPUR', the NPTEL logo, and the text 'NPTEL ONLINE CERTIFICATION COURSES'. A small video inset in the bottom right corner shows a man in a light blue shirt speaking.

Let see the different types of cohesion. We can look at a design and we can say that it has any one of these seven types of cohesion (below slide).

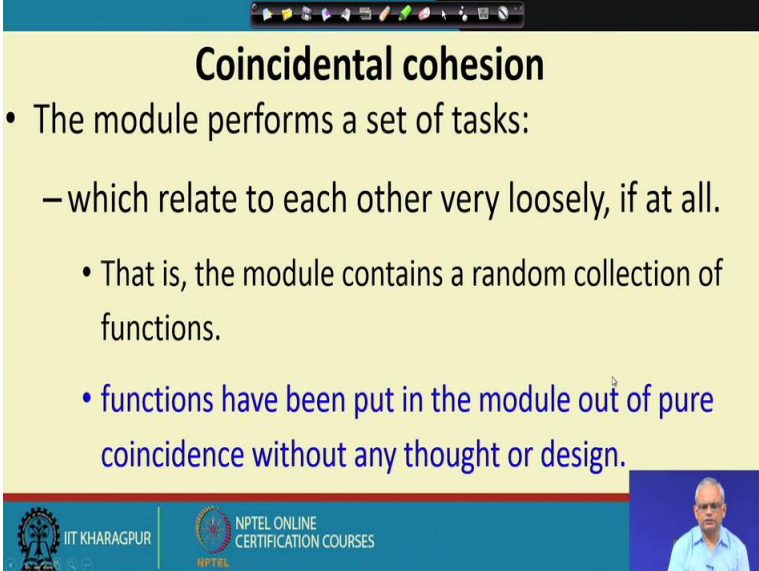
(Refer Slide Time: 22:32)



We can look at a module and we can say that this module has either this coincidental, logical, temporal, procedural, communicational, sequential or functional cohesion.

If, it has functional cohesion then it has high cohesion means it's a good design. But if it has coincidental cohesion on the other hand, it is not a good design. And, something in between functional and coincidental is moderate cohesion. We want to look at all modules and, see what type of cohesion are there. If most of them are high cohesion or a slightly moderate cohesion we will say that the design is good or ok. But if they have low cohesion, we will see that we will say that it's not a good design.

(Refer Slide Time: 23:41)



Coincidental cohesion

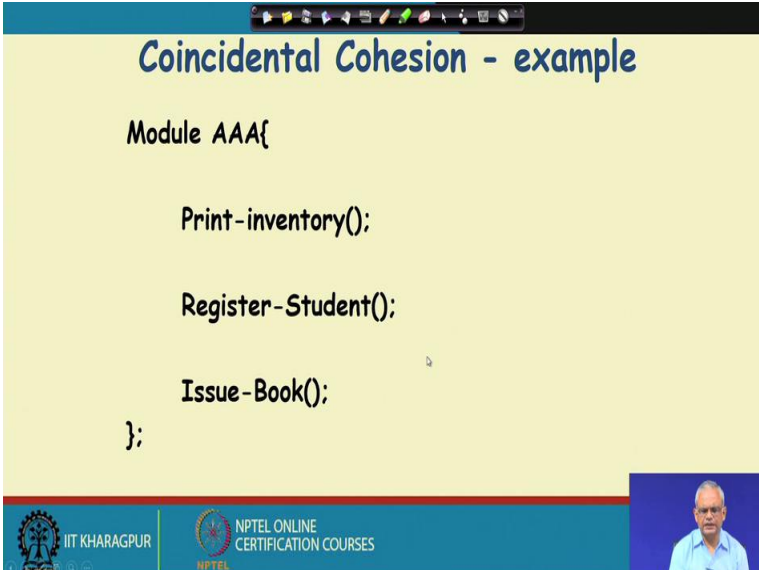
- The module performs a set of tasks:
 - which relate to each other very loosely, if at all.
- That is, the module contains a random collection of functions.
- functions have been put in the module out of pure coincidence without any thought or design.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

First let's look at the worst form of cohesion, it's called as the coincidental cohesion.

Here, if we look at the module for look through what functionality it has, we will see that the functions are rather random collection of the functions. There is no thinking or plan behind putting some functions in the module, we just taken out random functions and made them into a module.

(Refer Slide Time: 24:15)



Coincidental Cohesion - example

```
Module AAA{  
  
    Print-inventory();  
  
    Register-Student();  
  
    Issue-Book();  
};
```

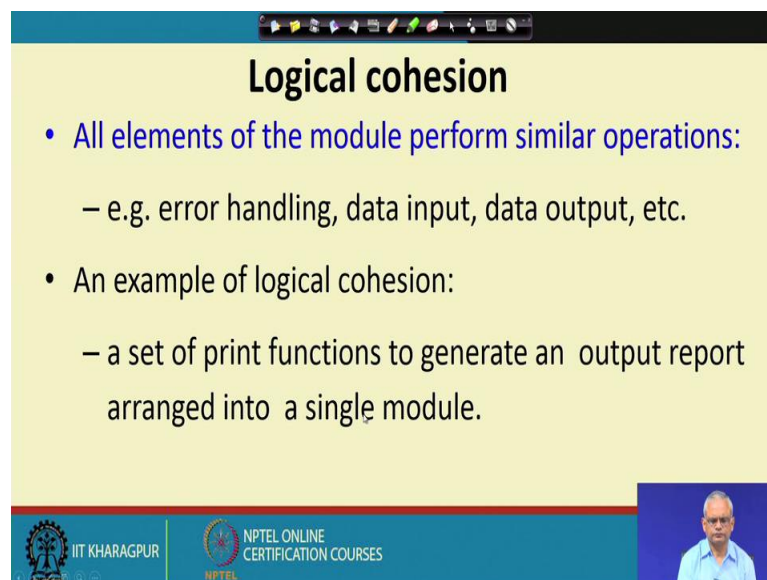
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Just to give an example, let say we look at one module named as AAA and then it contains the functions: print-inventory, register-student, issue-book.

So, this actually correspond to the functionalities required by various other, various requirements. Print-inventory() is part of some inventory functionality, Register-student() is part of some registration, Issue-Book() is part of library management system. So, these are random set of functions, they do not neither belong to the same requirement nor there is any relationship between these functions.

And, if we ask that what does the module AAA achieve? We cannot really answer. We will say that it has some part of the inventory, does some registering student, it also does issue book et cetera. So, there is no single thing, single function that this module achieves. It Does various things here and there. Somebody has just put these functions randomly into this module AAA and this we call as coincidental cohesion.

(Refer Slide Time: 25:45)



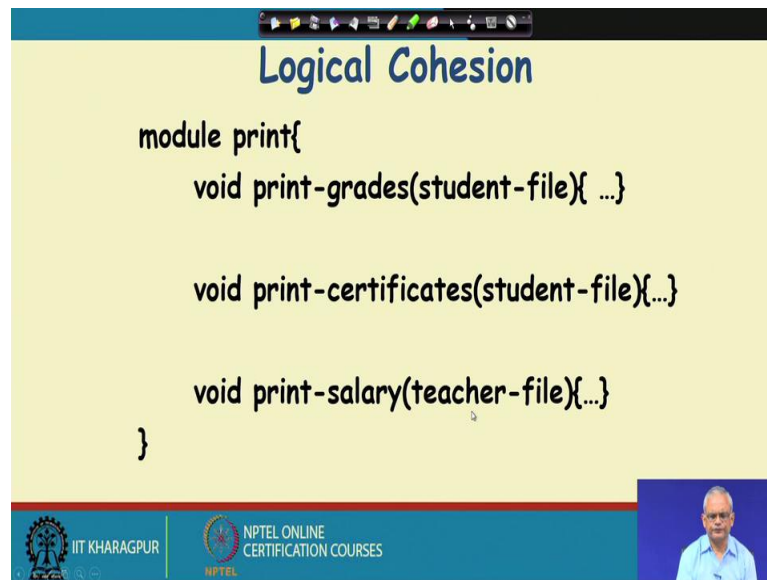
Logical cohesion

- All elements of the module perform similar operations:
 - e.g. error handling, data input, data output, etc.
- An example of logical cohesion:
 - a set of print functions to generate an output report arranged into a single module.

The slide is a presentation slide with a yellow background and a blue header. It contains two bullet points. The first bullet point is 'All elements of the module perform similar operations:' followed by an example 'e.g. error handling, data input, data output, etc.'. The second bullet point is 'An example of logical cohesion:' followed by an example 'a set of print functions to generate an output report arranged into a single module.'. In the bottom right corner, there is a small video inset showing a man in a blue shirt. The bottom of the slide features logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES.

We, consider logical cohesion module has logical cohesion, if all elements, all functions in the module performs similar operation. For example, all functions in a modular doing error handling or all functions in a modular reading the data input or all functions in a modular just print kind of thing, data output. Then, we say that this modular design has a logical cohesion that all the functions it has are logically related that they are doing similar things and that's why designer has put all the functions in a modular design. And, this logical cohesion is not a very good form of cohesion slightly better than coincidental cohesion, but still it's not good.

(Refer Slide Time: 26:43)



Logical Cohesion

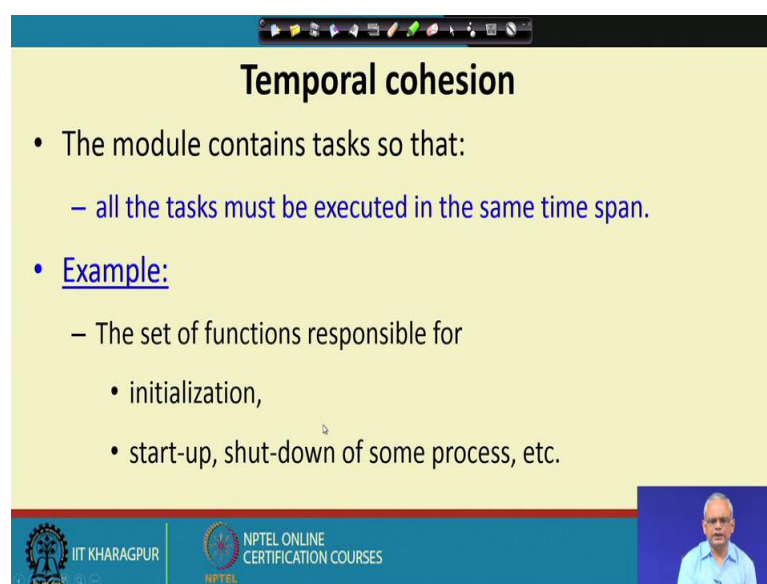
```
module print{  
    void print-grades(student-file){ ...}  
  
    void print-certificates(student-file){...}  
  
    void print-salary(teacher-file){...}  
}
```

The slide features a yellow background with a blue header and footer. The header contains the title 'Logical Cohesion' in a large, bold, blue font. The code snippet is written in black text. The footer includes the IIT Kharagpur logo and the NPTEL Online Certification Courses logo. A small video inset of a speaker is visible in the bottom right corner.

Let's look at an example of logical cohesion. Let say we have a module, name is print and then it does various types of printing, across various functionalities: printing grade, printing certificates, printing salary et cetera. All these printing functions are part of various other functionalities or requirements.

And, then we say that this module has logical cohesions, it's not the bottom most cohesion, but still it's not a good cohesion.

(Refer Slide Time: 27:23)



Temporal cohesion

- The module contains tasks so that:
 - all the tasks must be executed in the same time span.
- Example:
 - The set of functions responsible for
 - initialization,
 - start-up, shut-down of some process, etc.

The slide features a yellow background with a blue header and footer. The header contains the title 'Temporal cohesion' in a large, bold, blue font. The content is organized into a bulleted list. The footer includes the IIT Kharagpur logo and the NPTEL Online Certification Courses logo. A small video inset of a speaker is visible in the bottom right corner.

Next, we will look at the temporal cohesion. We will stop here and will continue in the next lecture.