

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

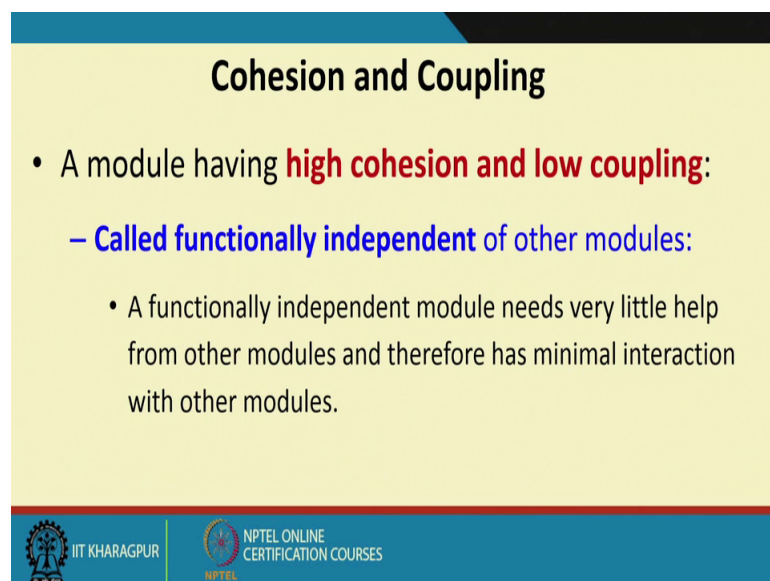
**Lecture – 21**  
**Classification of Cohesion**

Welcome to this lecture. In the last lecture, we were discussing some very general concepts about a good design. And, we had said that design is an iterative procedure and the designer can come up with different designs, but the different designs can vary in their quality.

And, we are trying to find out how to distinguish between 2 designs? When can I say that a design is good or bad? We identify some characteristics of a good design. One of the very important characteristics is modularity. The different modules in the designs should be more or less independent and we said that functional independence is an important characteristic in good design. Now, we said that to identify functional independence, we need to look at the cohesion and coupling among the modules.



Now, let us proceed from that point.

(Refer Slide Time: 01:42)



**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

A good design where the modules functionally independence, these have high cohesion and low coupling. If they have high cohesion and low coupling, we say that a modules are functionally independent, but then we must define the terms cohesion and coupling.

(Refer Slide Time: 02:18)

### Advantages of Functional Independence

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

The diagram illustrates two module structures. The top structure, labeled 'No dependencies', shows four blue rectangular modules arranged in a 2x2 grid with no arrows between them. The bottom structure, labeled 'Highly coupled-many dependencies', shows four blue rectangular modules arranged in a 2x2 grid with multiple arrows between them, indicating high coupling.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Having a module structure where the modules; so, functional independence has many advantages. One advantage is understandability; this is an example of modules where there are no independence no dependence.

So, the modules are independent and here we can understand the different modules one by one just check out this one module we can understand, there is no dependency. Whereas, here where we have these modules there is high dependency to be able to understand one module, we need to also understand the modules on which it depends.

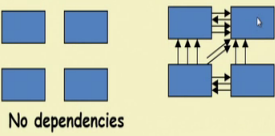
And, we can see that a module has dependency structure with many other modules. And therefore, to understand one module we will have to understand many other modules and task becomes extremely difficult, but if there is functional independence. We can easily understand the modules and also any error condition in a module is isolated. If, we find that the function a system has a whole is not a function there is a error. We can easily identify the error, but if they are dependent on each other heavily dependent on each other, we will have to keep on tracing the education sequences and it may lead to just going round and round without really able to determine the error.

So, the advantages of functional independence is that the complexity of the design is reduced, it becomes easy to maintain debugs and also we can take out any one module here and re use in another application. Whereas, here this situation is may not be possible because, if we want to take this module for re using, we might have to take the other modules on which it dependence or calls the functions of other modules.

(Refer Slide Time: 04:50)


**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.
- **Also: Reuse of modules is possible.**



No dependencies

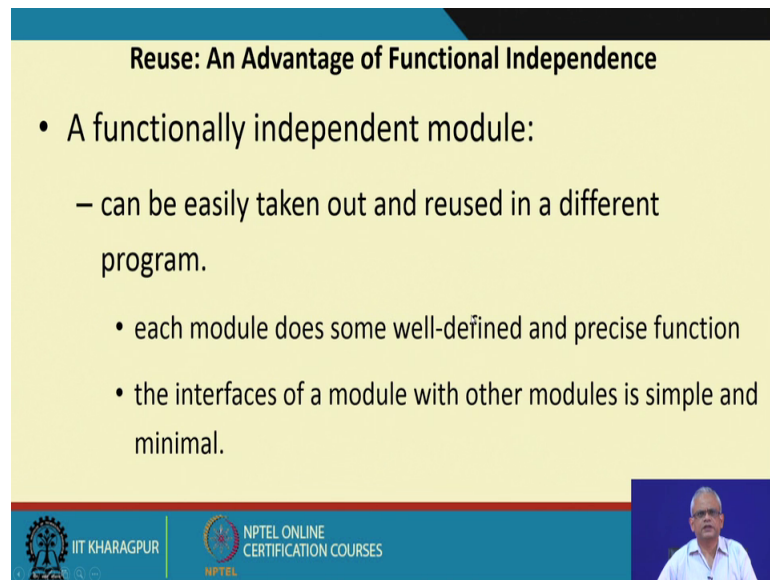
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES



Now, let us list the advantages of functional independence; one is that it reduces the complexity of the system. It makes easy to understand, it reduces error propagation any error in a module when the system fails; you can easily trace it to this module. Whereas if they are heavily dependence you can just go round and round trying to figure out where the error is, and becomes very difficult to debug if there is a high coupling and there is a heavy dependency among modules.

The third reason why functional independence is advantageous is that re use of modules becomes easier. If, we have developed an application consisting a functional in dependent modules, we can just check out any module and re use in another application without much problem. Where as if our modules structure is like this then you can just check out one module and use it if you take this, we will have to also take out that entire application.

(Refer Slide Time: 06:18)



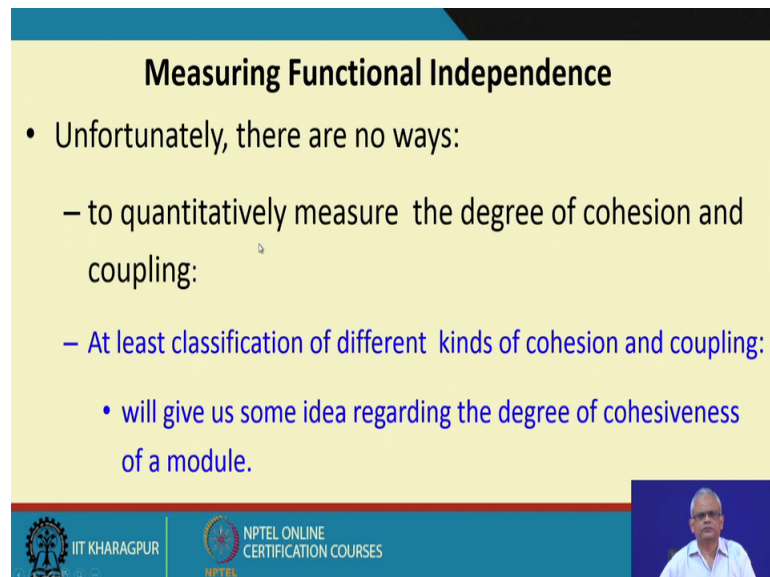
**Reuse: An Advantage of Functional Independence**

- 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

Reuse is a major issue in software engineering, because it reduces cost if we can reuse parts of one application, another application, it is very advantageous. And therefore, in the design stage, we have to design such that reuse is facilitated and that becomes possible if the module structure are functionally independent.

(Refer Slide Time: 06:55)



**Measuring 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

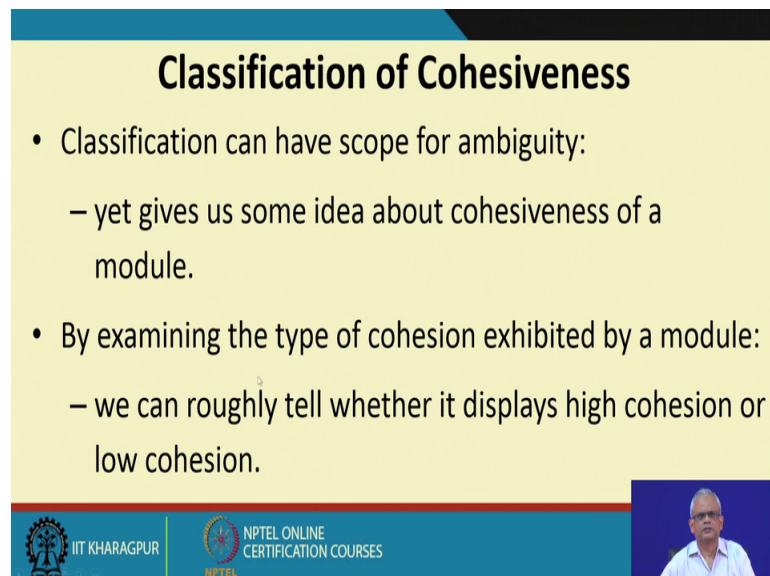
Now, we have been saying that functional independence is good it has many advantages and so on, but given a design structure it may not be that all modules are totally independent. That is a very idealistic design where no module ever interact with any

other module; in real applications modules do interact with each other they call each other's functions and so on.

But, the given an arbitrary application design can we measure the functional independence of that design? So, that we can say that we can say that this is a better design in another design. It would have been very nice if we could have come up with very quantitative measure of the degree of independence, but we can do the next thing it is very difficult to come up with quantitative measure, but we can do one thing is that, we can classify the cohesion and coupling and based on that, we can approximately say which is a better design than another.

So, let us me just repeat that point that quantitative measure of functional independence, that is the cohesion and coupling in terms of their cohesion and coupling. There is no accepted practice there is no accepted technique by which we can measure the functional independence and everybody will agree with that. But we can do the next best thing, that is we can characterize the cohesion and coupling in terms of certain classes. And, depending on the class to which an application belongs the cohesion and coupling; we can say that it is design is very good moderate bad and so on.

(Refer Slide Time: 09:07)




**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.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

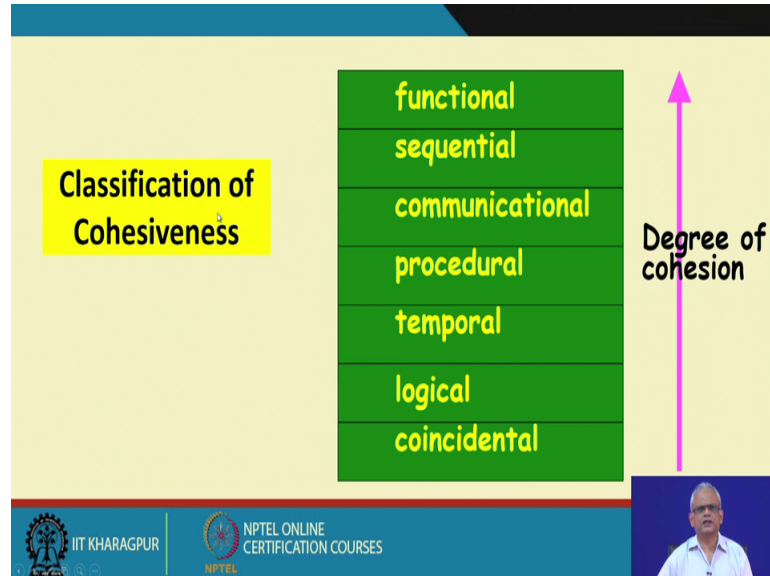
NPTEL



Let us see, how we can classify the cohesion and coupling existing in a design? Here, we will see that there is a small scope for ambiguity, because you can argue with that design belong to this class another class, but then more or less we get a fairly good picture. Even

though you do not quantitatively say that it is a 70.1 percent. We can say, that it is between 60 to 70 or between 70 to 80 and that it self we will help us to the great extent.

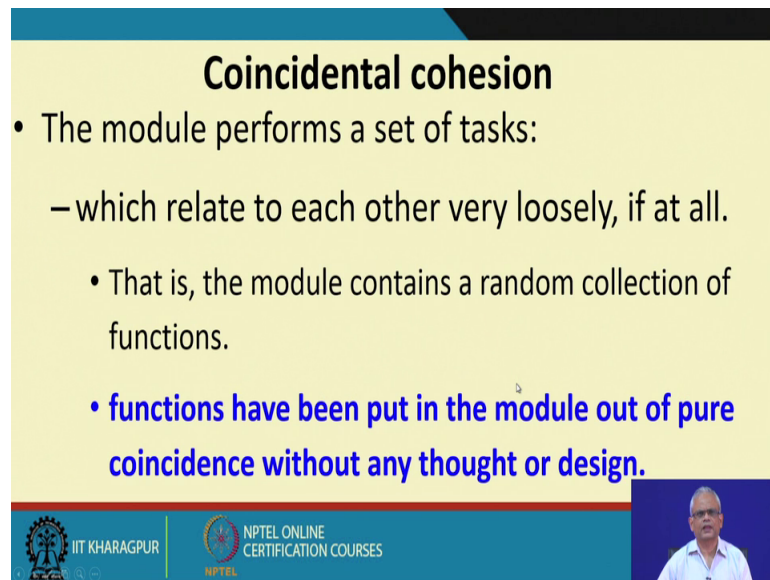
(Refer Slide Time: 09:55)



First let us look at cohesiveness, the cohesion existing in a module can be classified into co incidental, logical, temporal, procedural, communicational, sequential and functional. So, that is 7 classes. And, the worst form of cohesion is co incidental. You, can just examine a module and then you should be able to tell, where in this spectrum does the cohesion of the module lie, is it co incidental and this will be a bad form of cohesion or is it something like a temporal or procedural, which is middle order or is it functional, which is the best form of cohesion.

Now, let us see what are these different types of cohesion? So, that given a module structure we should be able to roughly place it in this spectrum.

(Refer Slide Time: 11:08)



**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 us see, the worst form of cohesion that is coincidental cohesion. Here, there is no thought or designed behind putting functions into a module, we are just randomly assigned them to modules. The different functions existing in the program they have been randomly put into different modules, without any thought or design.

If, this is the case that we select a set of functions, randomly and assign them to a module and we ask the question that can you tell us what does this module do? Will be very hard (Refer Time: 11:56) to give a simple answer to what the module does? We can say that it does these and these and these etcetera.

(Refer Slide Time: 12:06)

**Coincidental Cohesion - example**

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

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

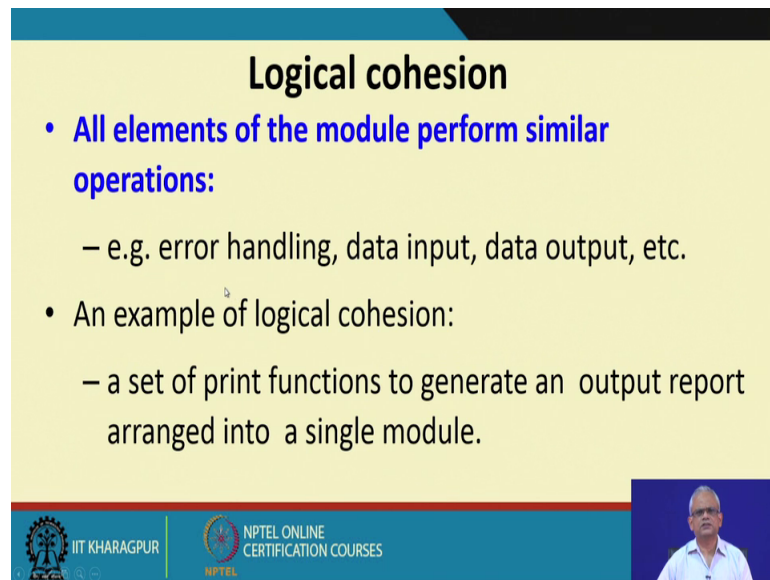
Let us look at an example. Let us say we have a module and we cannot even think of a good name to the module we just gave it the name AAA, because it does various stuff. In a library management software, there are many functions like register students, register book, collect fine, issue book, return book, and so on. We just took 3 functions here and just put them part of this module, that it does little bit of inventory print the current inventory and also register students and also issue book.

Now, if we ask what does this module do? We will say that it does some inventory functions like print inventory, it does some student registration and it also does issue book. So, we cannot give a simple one word description of this module to capture, what it does?

There are functions which are doing very different things. This is an example coincidental cohesion and given an example, you can easily find out given an example module you can easily find out whether it is a case of coincidental cohesion.



(Refer Slide Time: 13:53)



**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.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES




Now, let us look at the slightly better form of cohesion called as logical cohesion. Here all element here all elements of the module performs similar functions for example, all functions do error handling, all may do let us say print statements or read data scan statements. For example, all functions do error handling, all may do let us say print statements are read data scan statements are all print statements etcetera.

So, here all the functions here do similar functions and just because the functions do similar things we just decided to put them into one module. For example in an application, if we put all print functions that generate an output report into a single module then we say that the module has logical cohesion.

(Refer Slide Time: 14:46)

## Logical Cohesion

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






This cohesion also we can unambiguously and easily identify. For example, if we find module named print, which prints various things like it prints grades, it prints certificates, it prints salary slip, it prints inventory details and so on. So, just because all these functions do printing we decided to put them into one single module and call it print. And, then this is a logical cohesion not a very good form of cohesion, but better than coincidental or random cohesion.

(Refer Slide Time: 15:26)

## Temporal cohesion

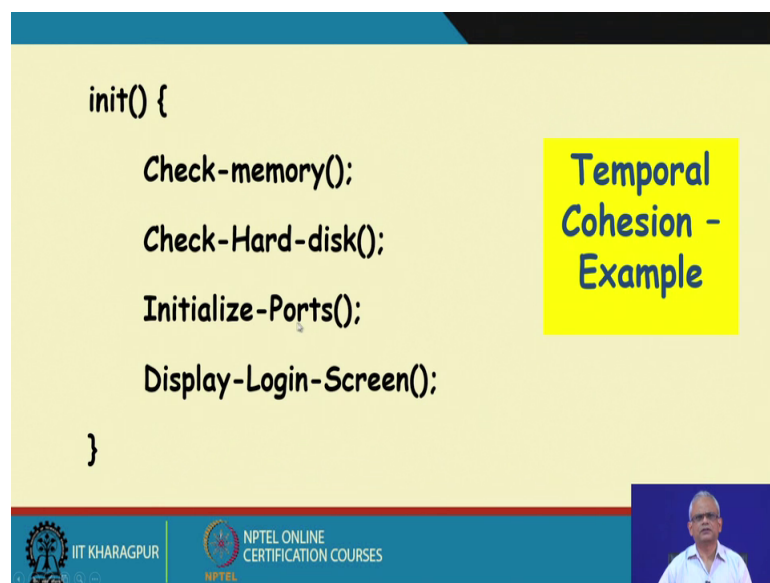
- The module contains functions so that:
  - **all the functions must be executed in the same time span.**
- Example:
  - The set of functions responsible for
    - initialization,
    - start-up, shut-down of some process, etc.



Slightly better form of cohesion is the temporal cohesion; here we group functions together if they are executed within the same time span.

For example, let us say during initialization certain functions are executed one after other and before any other function can execute the initialization functions done or let us say we have a certain shut down procedure etcetera. So, here these functions do very different things, but then they have been put into one module just because they run during the initialization time.

(Refer Slide Time: 16:17)



```
init() {  
    Check-memory();  
    Check-Hard-disk();  
    Initialize-Ports();  
    Display-Login-Screen();  
}
```

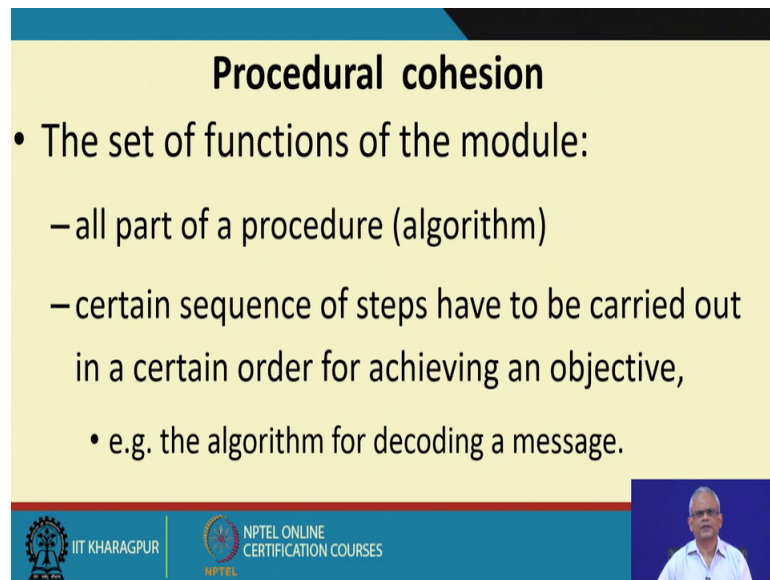
Temporal Cohesion - Example

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Just to give an example, let us say we have a function called `init` and then we do check memory, check hard disk, and then initialize the ports, then display some login prompt on the screen.

So, here the functions have very different purposes: something displays on the screen, something initializes a port, then checking whether memory is alright, hard disk is alright, etcetera, but then they are run on the same time span and that is the reason we have decided to put them in one module called `init`. And, this is an example of temporal cohesion that is the execution in the same time span.


(Refer Slide Time: 17:04)



**Procedural cohesion**

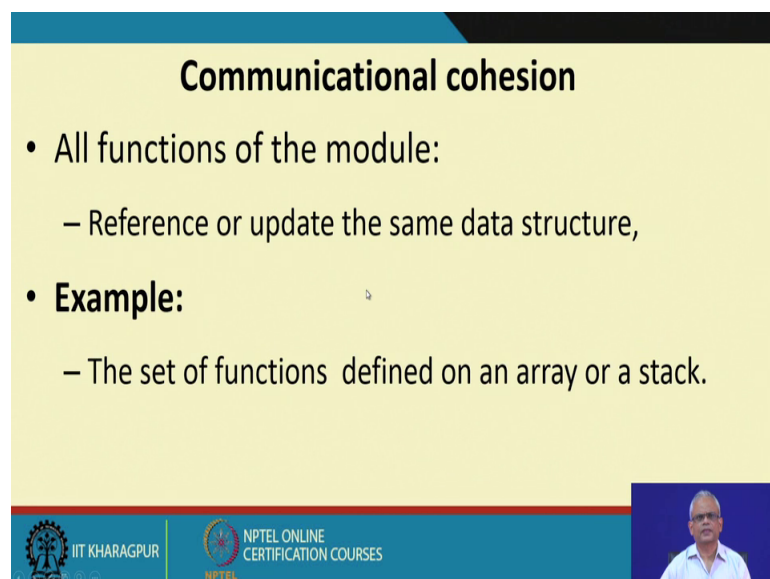
- The set of functions of the module:
  - all part of a procedure (algorithm)
  - certain sequence of steps have to be carried out in a certain order for achieving an objective,
    - e.g. the algorithm for decoding a message.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES



A still better form of cohesion is the procedural cohesion. Here the different functions in the module their part of some algorithm. For example, that they carry out some steps, let us say for decoding a message, they do some steps like initialize and then let us say discrete cosine transfer and then let us say entropy calculation and so on. So, these are set of steps, which is part of algorithm and we have just put them together, but because they are part of the algorithm in they do very different things.


(Refer Slide Time: 18:01)



**Communicational cohesion**

- All functions of the module:
  - Reference or update the same data structure,
- **Example:**
  - The set of functions defined on an array or a stack.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES





A better form of cohesion is communicational cohesion. Here the different functions they operate on the same data structure. So, the result is shared between the different functions. Just consider the set of functions defined on a array or stack, let us say stack push pop etcetera. So, here the functions on the stack they all operate on the stack and use the data structure. And, this we can call it as a communicational cohesion.

(Refer Slide Time: 18:41)

### Communicational Cohesion

```
handle-Student- Data() {  
    Static Struct Student-data[10000];  
    Store-student-data();  
    Search-Student-data();  
    Print-all-students();  
};
```

Communicational  
Access same data

 IIT KHARAGPUR |  NPTEL ONLINE CERTIFICATION COURSES


These an example of a communicational cohesion is handle student data is the name of the module. And, then there is a data here student data 10 000. So, this is the data here. And, then you have various functions there in the module, which are operating on this. For example, search student data print all data and store student data update student data and so on.

So, this is a better form of cohesion than the cohesion that we talked about so far is the communicational cohesion.

(Refer Slide Time: 19:24)

### Sequential cohesion


- Elements of a module form different parts of a sequence,
  - output from one element of the sequence is input to the next.
  - Example:



```
graph TD; A[sort] --> B[search]; B --> C[display];
```

The diagram illustrates sequential cohesion with three green boxes containing the words 'sort', 'search', and 'display' stacked vertically. Downward-pointing arrows connect 'sort' to 'search' and 'search' to 'display', indicating a sequential flow of data or operations.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES




And a still better form of cohesion is a sequential cohesion where the different functions in the module share data, but then they just do not randomly update the data here the data is passed from one function to the other in a sequence. For example, first do sorting then, search then, display and so on.

(Refer Slide Time: 19:51)

### Functional cohesion

- Different elements of a module cooperate:
  - to achieve a single function,
  - e.g. managing an employee's pay-roll.
- When a module displays functional cohesion,
  - **we can describe the function using a single sentence.**

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES



But, the best form of cohesion is a functional cohesion. Here, the different functions share the data, but then they work towards achieving a single function. For example, if all the functions to manage the employees' pay roll are just put together into a module,

like compute overtime, compute the current months' pay, change the basic salary and so on.

So, these are all towards managing the employee's payroll. And, here one of the distinguishing characteristics is that by looking at the module structure, you can just give a very simple name here to the module saying that manage employees pay roll. Because all functions work towards doing some parts of the managing employees payroll. So, one test whether module has functional cohesion is that we should be able to describe the function of the module or what the module achieves by using a very simple sentence?

(Refer Slide Time: 21:23)

Write down a sentence to describe the function of the module

**Determining Cohesiveness**

- If the sentence is compound,
  - it has a sequential or communicational cohesion.
- If it has words like “first”, “next”, “after”, “then”, etc.
  - it has sequential or temporal cohesion.
- If it has words like initialize,
  - it probably has temporal cohesion.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, let us see how to identify the cohesiveness of a module? Let us say we are given a module and we are asked to find that, what is the cohesion here? Is it a good form of cohesion, bad form of cohesion?

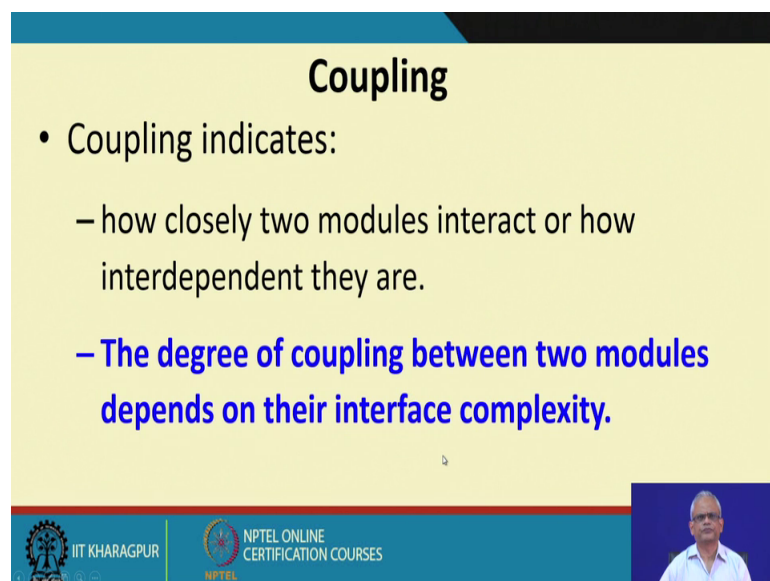
Now we know the 7 classes of cohesion starting from very bad case of coincidental cohesion. And, then the best form of cohesion that is the functional cohesion. We, can imagine that there may be some ambiguity, whether to consider it as a sequential cohesion or let us say procedural cohesion and so on, but then if it is a bad case of cohesion, we can easily identify or whether it is a somewhat level of, but then or a very good case of, but then cohesion so, that much we can easily identify.

One hint about the identifying the cohesion is that first look at the module and the functions that it has and try to describe what is the function of the module? What does the module achieve? And, then we write this in a sentence and if we find that if the sentence is compound that it does the this and this and so on, then it has a sequence or communication cohesion.

We are not mentioning here the coincidental cohesion, because that is easily identified and that is a bad case of cohesion, that the functions are totally unrelated, but these are the middle cases here we are just trying to give some hint. How to identify? If, it is a compound sentence, then we can suspect that it has sequential or communicational cohesion.

If, the sentence was worst like first it does these it does these and afterwards it does this and so on. Then it is either a sequential or temporal cohesion. If it has words like the initialize shut down procedure etcetera, then it is possibly has temporal cohesion.

(Refer Slide Time: 24:03)



**Coupling**

- Coupling indicates:
  - how closely two modules interact or how interdependent they are.
  - **The degree of coupling between two modules depends on their interface complexity.**

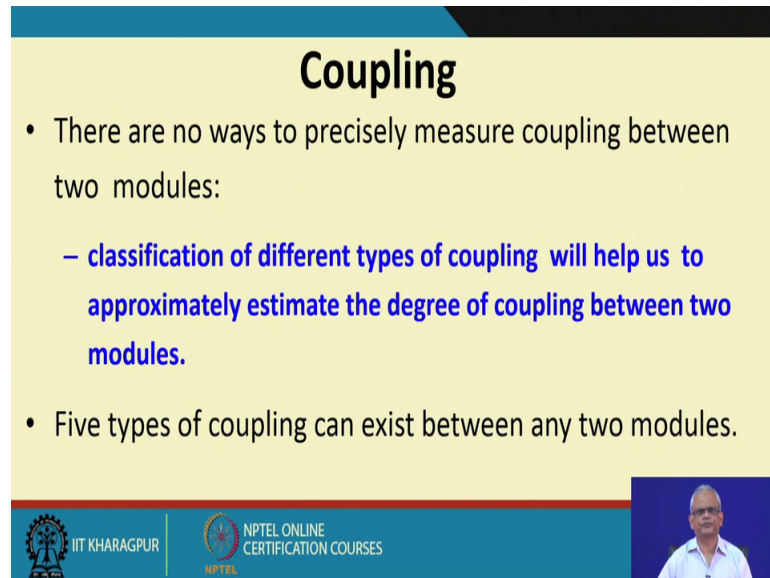
The slide features a yellow background with a blue header and footer. The footer contains the IIT Kharagpur logo and the NPTEL Online Certification Courses logo. A small video inset of a speaker is visible in the bottom right corner.

So, far we looked at the cohesion classification and given a module how to identify the cohesiveness? Now, let us look at coupling given 2 modules can we identify what is the extent of coupling between these 2 module. We say that 2 modules are highly coupled, if they have a very complex interaction among each other. If, there is no interaction, then will say that there is no coupling, but then the cases where the interacts that is the call one module calls function of another module. Then will say that there is a coupling, but



then we will see how to identify the class of coupling or how complex is the coupling? Roughly, we can say that the coupling between 2 modules can be identified by looking at the interface complexity.

(Refer Slide Time: 25:11)



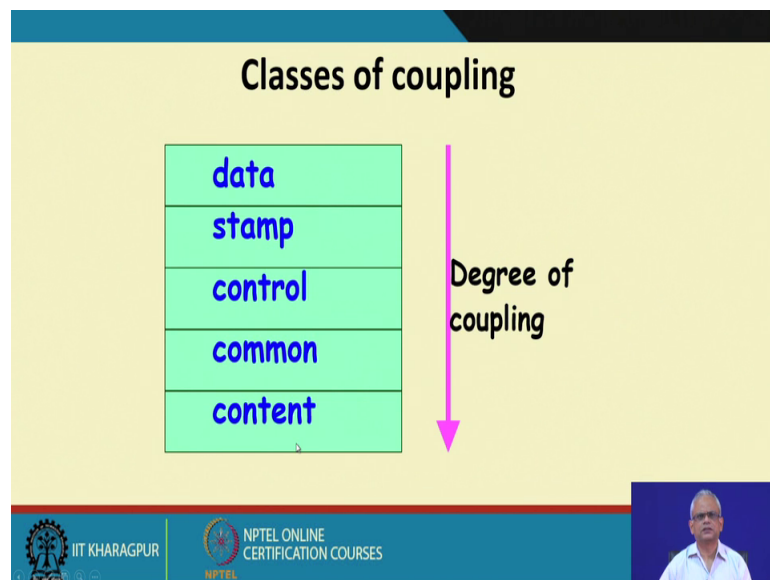
**Coupling**

- There are no ways to precisely measure coupling between two modules:
  - **classification of different types of coupling will help us to approximately estimate the degree of coupling between two modules.**
- Five types of coupling can exist between any two modules.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, just like we did the classification or cohesion, we will do a classification of coupling, there are 5 types of coupling it exists.

(Refer Slide Time: 25:23)



**Classes of coupling**

data
stamp
control
common
content

Degree of coupling

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, these are named as data coupling, stamp coupling, control coupling, common coupling, and content coupling. The data coupling is a simple form of coupling and it is a

low coupling and it is a good case of coupling. And, if it is a stamp coupling then there is a complex data that is interchanged in simple data coupling, only simple data items like integer or character etcetera simple data items are exchanged between 2 modules.

In stamp coupling more complex data items like a array or a big structure etcetera are exchanged between different modules. And, this is still an form of coupling worst form of coupling and control coupling, where one module decides the control path in another module. Common coupling they share some common data and then the very bad case of coupling is content coupling and this is so bad content coupling, that earlier using assembly routines and machine language programming content coupling was possible, but now all high level languages they it is difficult to write a program where there will be content coupling.

So, the bad cases the worst case of content coupling they cannot even write that code now in high level languages, because the high level languages have not designed such that this coupling does not occur, because it is a very bad case of coupling and make a program extremely complex. We will stop at this point and we continue in the next lecture and we look at the how to given a module 2 modules? Can we study these 2 modules and identify, what is the extent of coupling or which class of coupling out of this 5 classes exists between those 2 modules?

We will stop now and continue in the next lecture.

Thank you.