

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

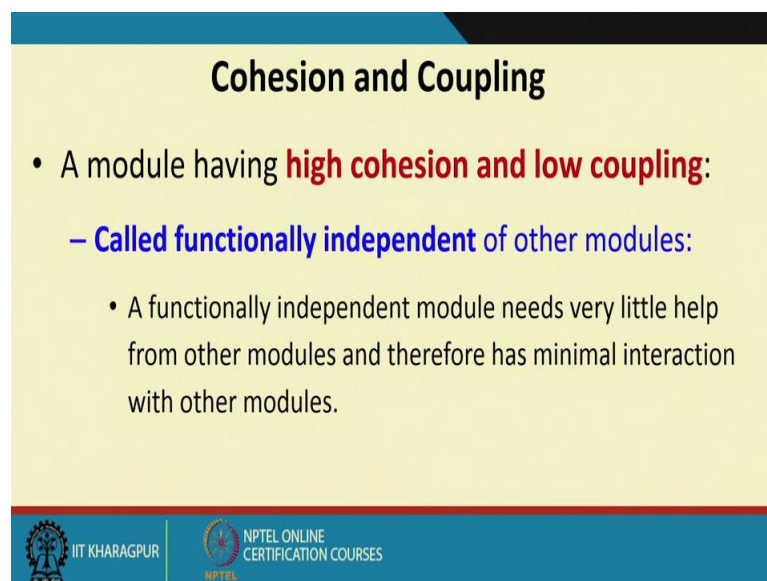
Lecture – 21
Classification of Cohesion

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 two designs? When can we say that a design is good or bad? We identify some characteristics of a good design. One of the very important characteristic 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's 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

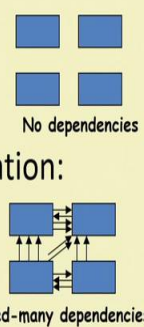
A good design where the modules are functionally independent, these have high cohesion and low coupling. If they have high cohesion and low coupling, we say that a

module is 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 scenarios of module dependencies. The top scenario, labeled 'No dependencies', shows four blue rectangular modules arranged in a 2x2 grid with no connecting lines between them. The bottom scenario, labeled 'Highly coupled-many dependencies', shows four blue rectangular modules arranged in a 2x2 grid with numerous arrows indicating complex interdependencies between all modules.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

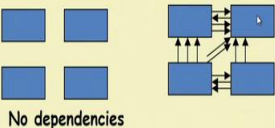
Functional independence has many advantages. One advantage is understandability. This is an example (on the above slide) of modules where there is no dependence. So, the modules are independent and here we can check one by one, we can understand, there is no dependency. Whereas, in another modules there is high dependency. And, here we can see that a module has dependency structure with many other modules. And therefore, to understand one module 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 an error in a functional independence module, we can easily identify the error. But if they are dependent on each other heavily, we will have to keep on tracing the 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, in the dependent module all these 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.**



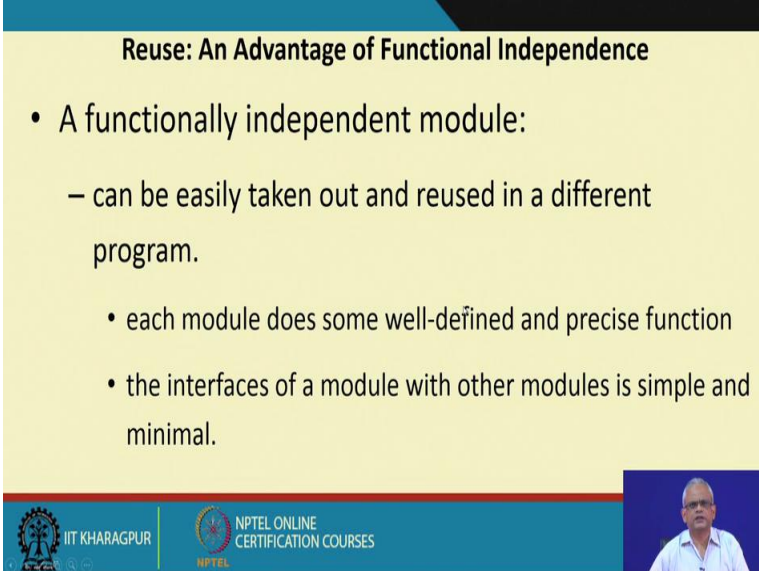
No dependencies

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES



Now, let's list the advantages of functional independence: one is that it reduces the complexity of the system, so it makes easy to understand the system. Another advantage is that 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 dependent and if there is high coupling, you can just go round and round trying to figure out where the error is, and becomes very difficult to debug. The third reason why functional independence is advantageous is that re use of modules becomes easier. If, we have developed an application consisting of functional independent modules, we can just take out any module and re use in another application without much problem. Where as if our modules structure is like highly dependent module then for taking one module, we will may have to also take out the entire application.

(Refer Slide Time: 06:18)



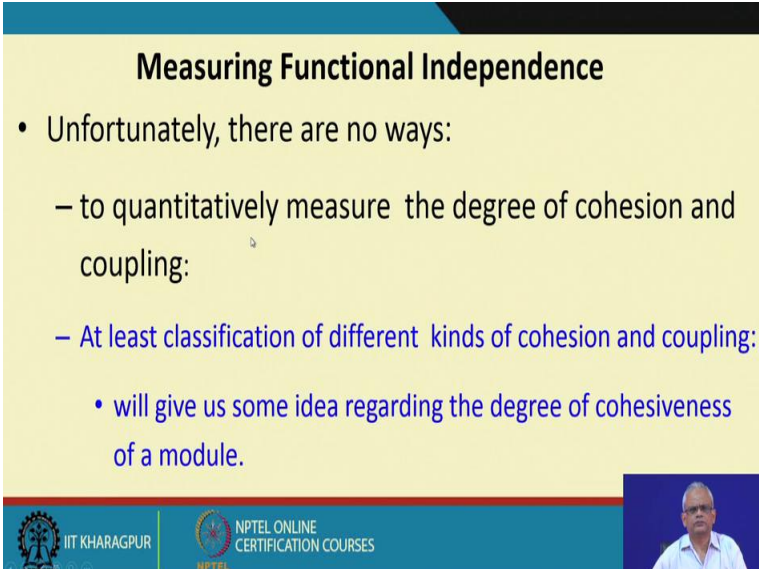
Reuse: An Advantage of Functional Independence

- 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. 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 is 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

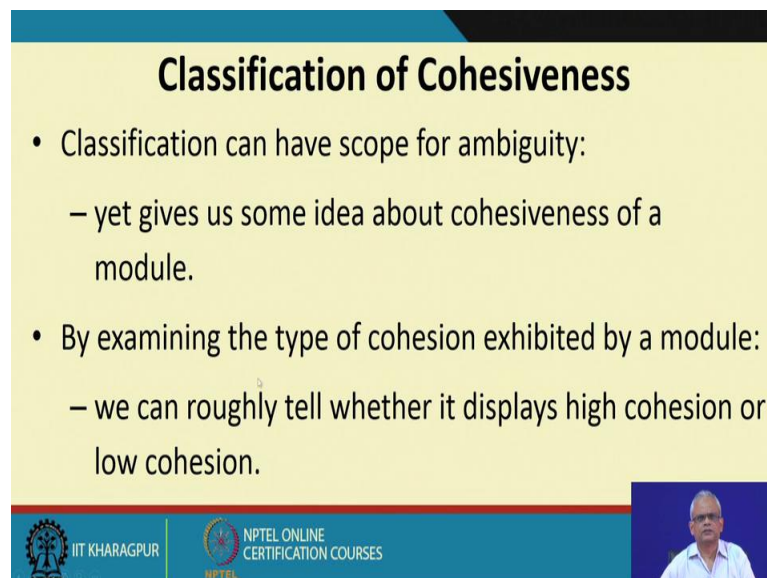
Now, we have been saying that functional independence is good, it has many advantages and so on. But given a design structure it may not be that all modules are totally independent. That's a very idealistic design where no module ever interacts with any

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

But then given an arbitrary application design can we measure the functional independence of that design? So, after measuring that we can say that one is a better design than another design. It would have been very nice if we could have come up with very quantitative measure of the degree of independence. 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 me just repeat that point that quantitative measure of functional independence 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, on the basis of that we can say a design is very good, moderate or bad.

(Refer Slide Time: 09:07)



Classification of Cohesiveness

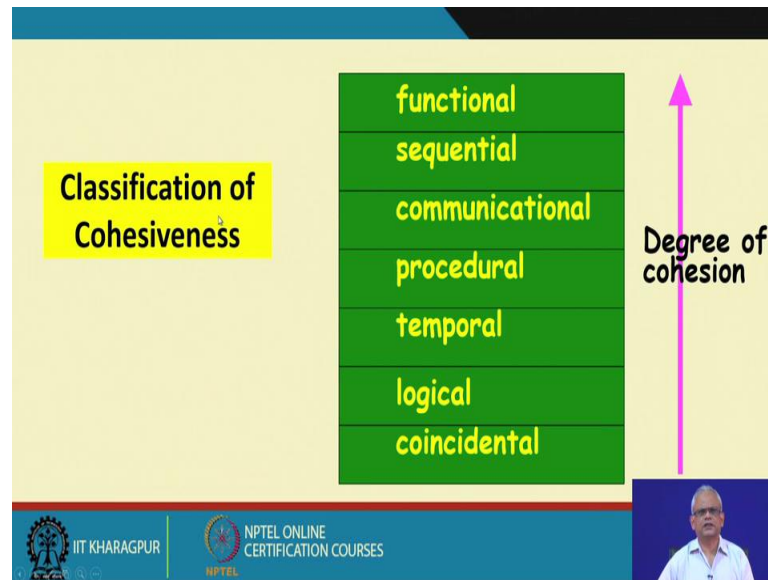
- Classification can have scope for ambiguity:
 - yet gives us some idea about cohesiveness of a module.
- By examining the type of cohesion exhibited by a module:
 - we can roughly tell whether it displays high cohesion or low cohesion.

The slide features the IIT Kharagpur logo on the left and the NPTEL Online Certification Courses logo on the right. A small video inset of a speaker is visible in the bottom right corner.

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 that design belong to some particular class or some another class, we get a fairly good picture about

cohesiveness of a module. For example, based on the cohesive class of a design, we 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 itself will help us to the great extent.

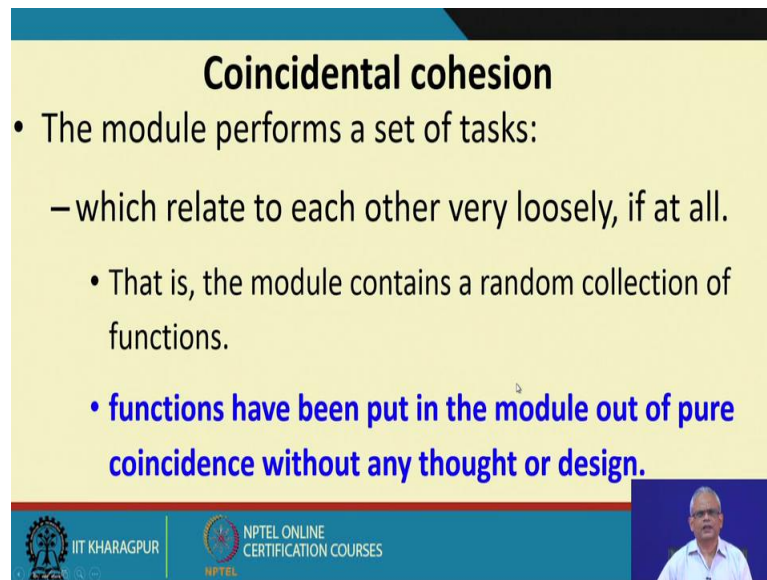
(Refer Slide Time: 09:55)



Now, look at cohesiveness. The cohesion existing in a module can be classified into coincidental, logical, temporal, procedural, communicational, sequential and functional. So, that is seven classes. And, the worst form of cohesion is coincidental. 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 coincidental (bad form of cohesion) or is it something like a temporal or procedural (middle form of cohesion) or is it functional (best form of cohesion).

Now, let 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)



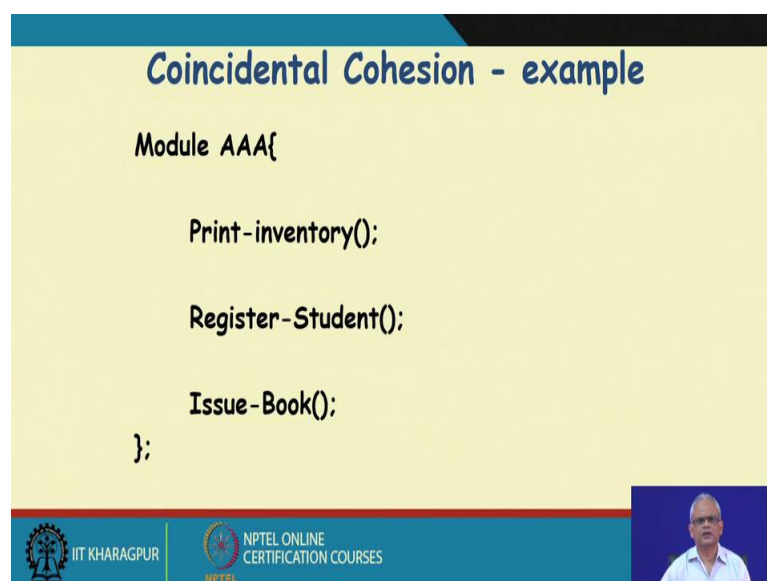
Coincidental cohesion

- 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 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 functions in 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 just select a set of functions, randomly and assign them to a module then it is coincidental cohesion and we ask the question that can you tell us what does this module do? Will be very hard pressed to give a simple answer to what the module does? We can say that it does this and this et cetera.

(Refer Slide Time: 12:06)



Coincidental Cohesion - example

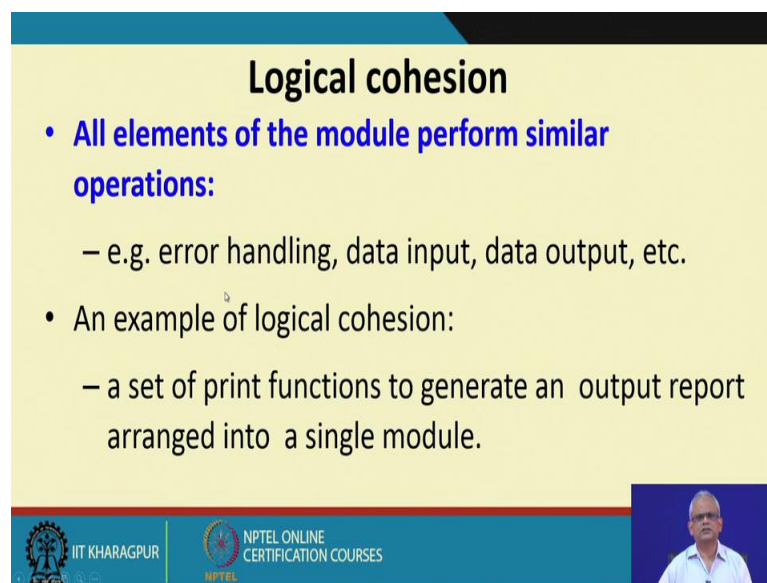
```
Module AAA{  
  
    Print-inventory();  
  
    Register-Student();  
  
    Issue-Book();  
};
```

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Let's look at an example. Let say we have a module named as AAA. 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. So, here are functions which are doing very different things. This is an example of coincidental cohesion and given an example, you can easily find out whether it's a case of coincidental cohesion or not.

(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.

The slide features a yellow background with a blue header and footer. The footer contains the IIT Kharagpur logo, the NPTEL Online Certification Courses logo, and a small video inset of a man in a white shirt.


Now, let's look at the slightly better form of cohesion called as logical cohesion. Here all elements of the module performs similar functions for example, all functions do error handling, all may do let say, print statements or read data, scan statements or all print statements et cetera.

So, here all the functions do similar things 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 will 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){...}  
}
```



IIT KHARAGPUR



NPTEL ONLINE
CERTIFICATION COURSES




This cohesion also we can unambiguously and easily identify. For example, if we find a 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.



IIT KHARAGPUR



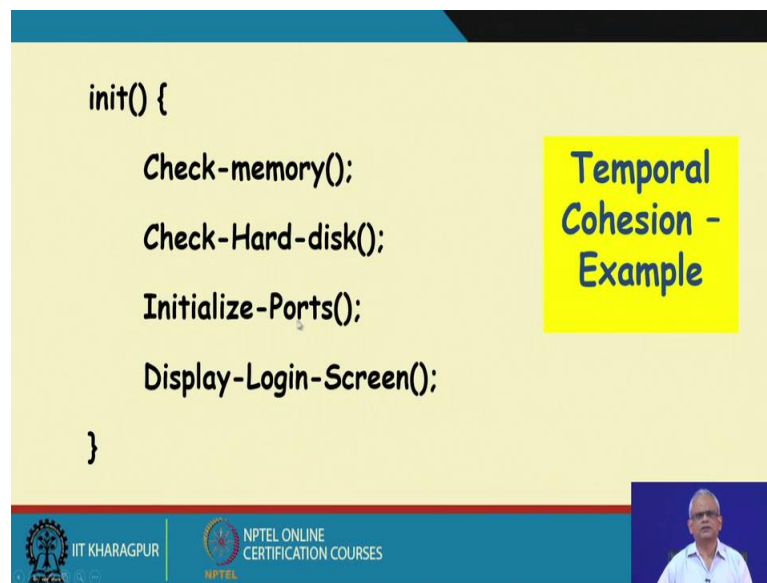
NPTEL ONLINE
CERTIFICATION COURSES



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 say during initialization certain functions are executed one after other and before the initialization any other functions executed or let say, we have a certain shut down procedure et cetera. 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)



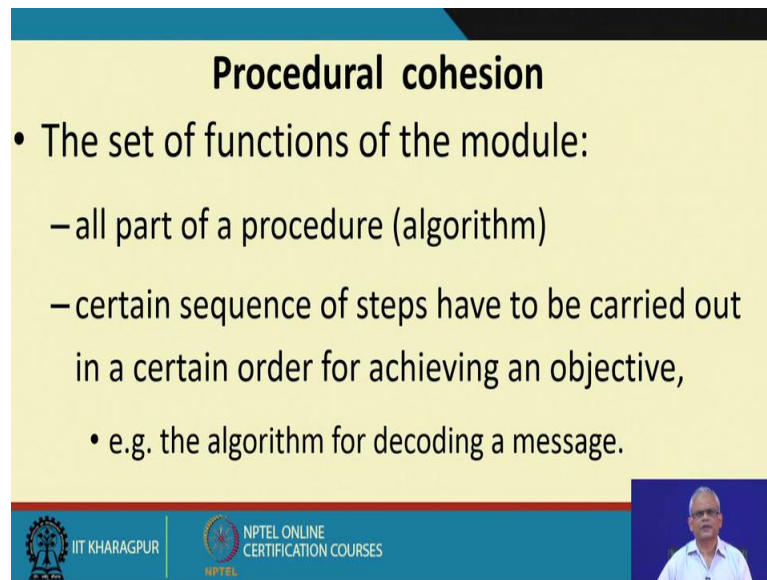
```
init() {  
    Check-memory();  
    Check-Hard-disk();  
    Initialize-Ports();  
    Display-Login-Screen();  
}
```

Temporal Cohesion - Example

The slide features a yellow background for the code area. A yellow box on the right contains the title 'Temporal Cohesion - Example'. The footer includes the IIT Kharagpur logo, the NPTEL logo, and the text 'NPTEL ONLINE CERTIFICATION COURSES'. A small video inset of a speaker is visible in the bottom right corner.

Just to give an example, let say we have a function called as `init()` and then we do check memory, check hard disk, then initialize the ports and then display some login prompts on the screen. So, here the functions are for very different purpose: some displays on the screen some message, some initialize port, then checking whether memory is alright, hard disk is alright, et cetera, 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 as `init()`. So, this is an example of temporal cohesion and here functions are execute in the same time span.

(Refer Slide Time: 17:04)



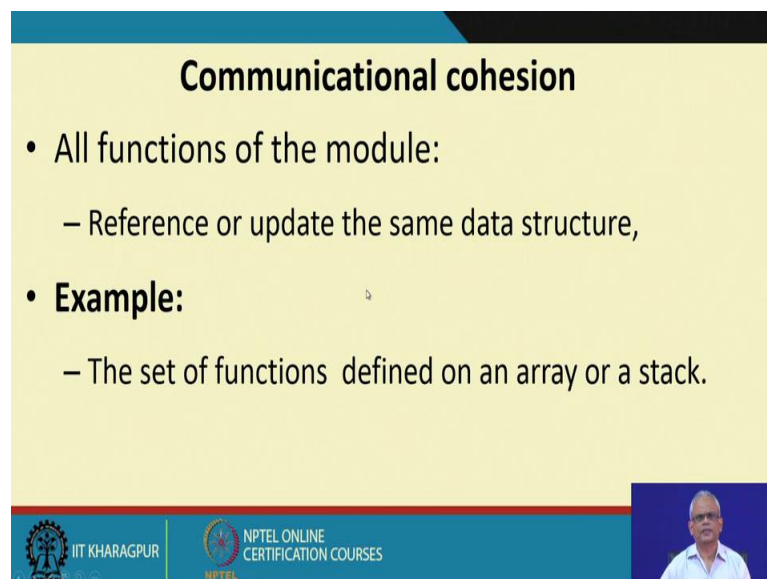
Procedural cohesion

- 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.

The slide 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 white shirt speaking.

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

(Refer Slide Time: 18:01)



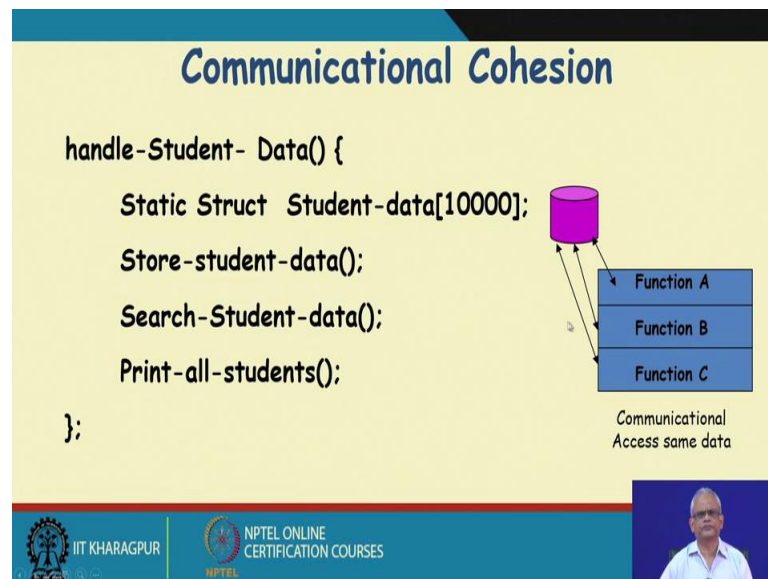
Communicational cohesion

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

The slide 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 white shirt speaking.

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

(Refer Slide Time: 18:41)




This an example of a communicational cohesion is handle-student-data is the name of the module. And, then there is an array here Student-data. The size of the array is 10000. So, this is the data here. And, then we 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 slide features a yellow background with a blue header and footer. The header contains the title 'Sequential cohesion'. The main content area lists a definition and an example. To the right of the text is a diagram with three green boxes labeled 'sort', 'search', and 'display' stacked vertically, connected by downward-pointing arrows. The footer includes the IIT Kharagpur logo, the NPTEL Online Certification Courses logo, and a small video inset of a speaker.

And a still better form of cohesion is a sequential cohesion. Here also 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

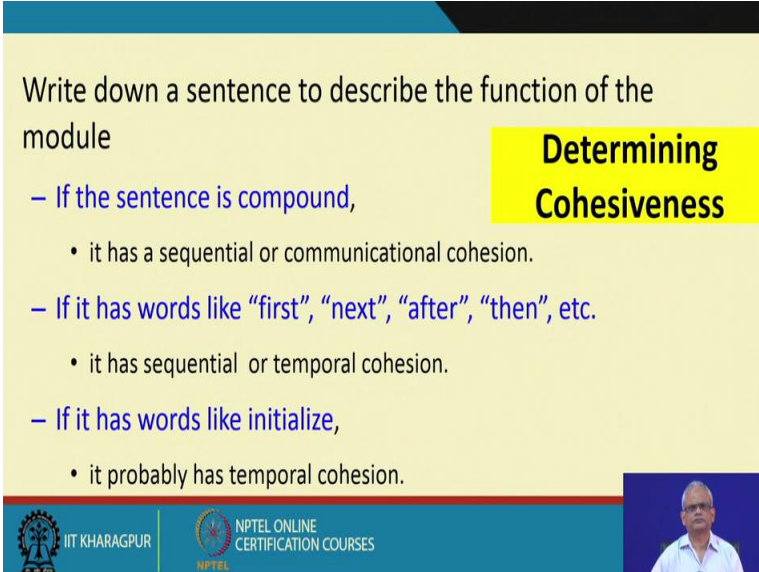
- Different elements of a module cooperate:
 - to achieve a single function,
 - e.g. managing an employee's pay-roll.
- When a module displays functional cohesion,
 - **we can describe the function using a single sentence.**

The slide features a yellow background with a blue header and footer. The header contains the title 'Functional cohesion'. The main content area lists two bullet points. The second bullet point has a sub-point in bold blue text. The footer includes the IIT Kharagpur logo, the NPTEL Online Certification Courses logo, and a small video inset of a speaker.

Now we will discuss the best form cohesion that is a functional cohesion. Here, the different functions they shared data, but then they work towards achieving a single function. For example, if all the functions to manage the employees pay roll, we just put

them together into a module like compute overtime, compute the current months' pay, change the basic salary everything put into a module. 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, we can give a very simple name. Here we give a name like 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's see how to identify the cohesiveness of a module? Let 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 or what?

Now, we know the seven 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 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 somehow ok level of cohesion or a very good case of cohesion we can easily identify.

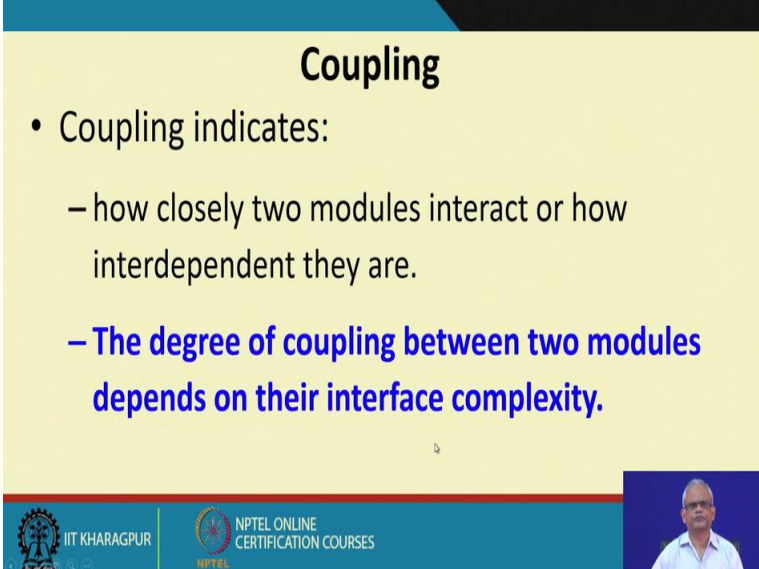
One hint about the identifying the cohesion is that first look at the module and the functions that it has and try to describe what is the function of the module? What does

the module achieve? And, then we write this in a sentence and if we find that if the sentence is compound sentence that it does this and this and so on, then it has a sequential or communicational cohesion.

We are not mentioning here the coincidental cohesion, because that's easily identified and that's 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 words like first it does these, then 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 et cetera, then it is possibly temporal cohesion.

(Refer Slide Time: 24:03)



Coupling

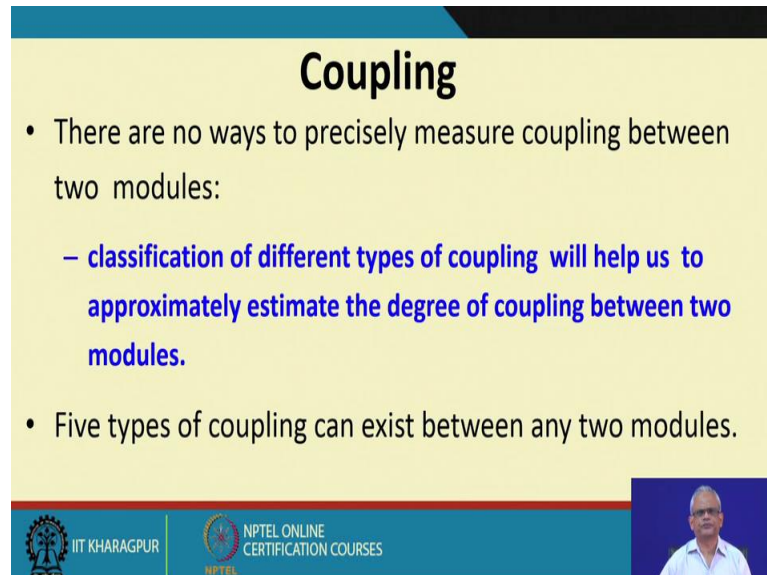
- Coupling indicates:
 - how closely two modules interact or how interdependent they are.
 - **The degree of coupling between two modules depends on their interface complexity.**

The slide features a yellow background with a blue header and footer. The footer contains the IIT Kharagpur logo, the NPTEL Online Certification Courses logo, and a small video inset of a man in a white shirt.

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

we can say that the coupling between two modules can be identified by looking at the interface complexity.

(Refer Slide Time: 25:11)



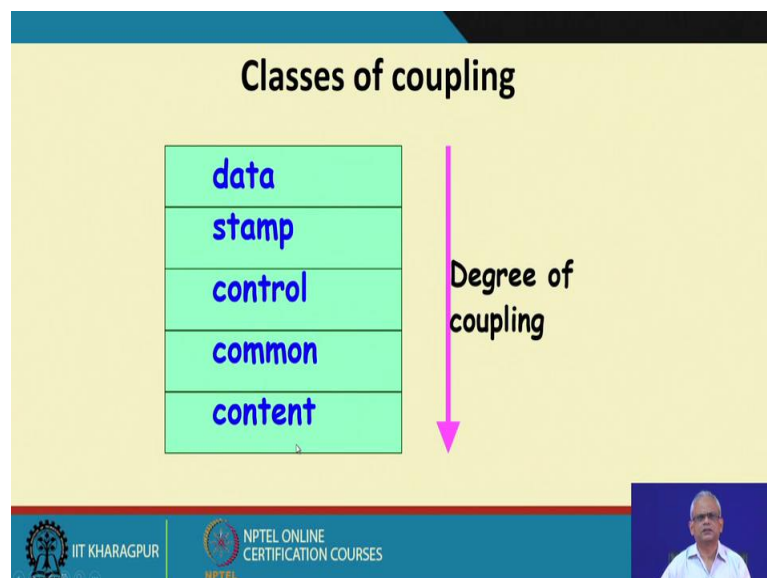
Coupling

- 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 five types of coupling exists:

(Refer Slide Time: 25:23)



Classes of coupling

data
stamp
control
common
content

Degree of coupling

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

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's a good case of coupling. And, if it is a stamp coupling then there is complex data

interchanged between module. In simple data coupling, only simple data items like integer or character et cetera simple data items are exchanged between two modules.

In stamp coupling, more complex data items like an array or a big structure, list et cetera are exchanged between different modules. And, this is still an ok form of coupling. Worse form of coupling is control coupling, where one module decides the control path in another module. In common coupling, they share some common data and then the very bad case of coupling is content coupling. One fact about content coupling is that earlier assembly routines and machine language programming content coupling was possible, but now all high-level languages, it is difficult to write a program where there will be content coupling. So, the content coupling code can be written in high-level languages, because the high-level languages have been designed such that this coupling does not occur, because it is a very bad case of coupling and make the program extremely complex.

We will stop at this point and will continue in the next lecture and we look at for a given module or two modules how to identify, what is the extent of coupling or which class of coupling out of these five classes exists between those two modules?

We will stop now and continue in the next lecture.

Thank you.