Software Engineering

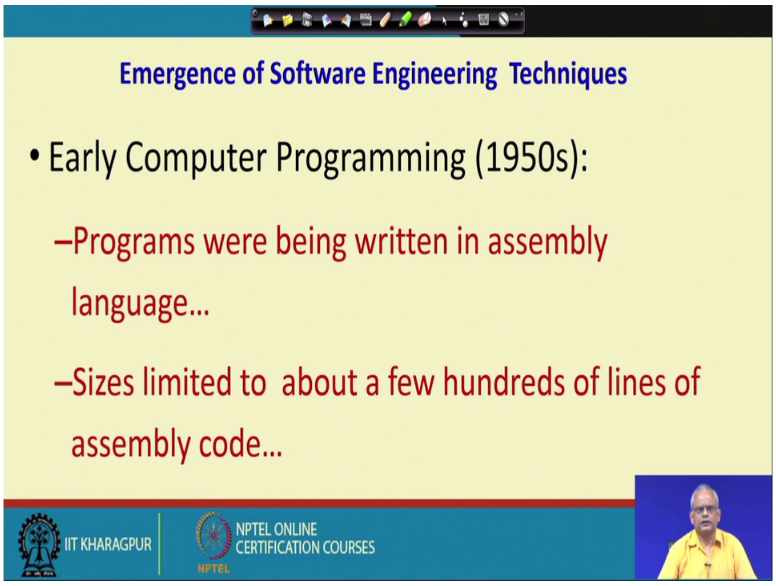
Prof. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 05
Introduction- V

Welcome to this lecture. Over the last few lectures we had looked at some introductory issues on software engineering and we are just starting to discuss how the software engineering techniques have evolved. Before we start looking at the software engineering principles it may be a good idea to just discuss briefly how these techniques have evolved over time; that will give us a better understanding about how the techniques have come into being.

Let us look at the evolution of the software engineering techniques starting from the early days of computer till now.

(Refer Slide Time: 01:16)



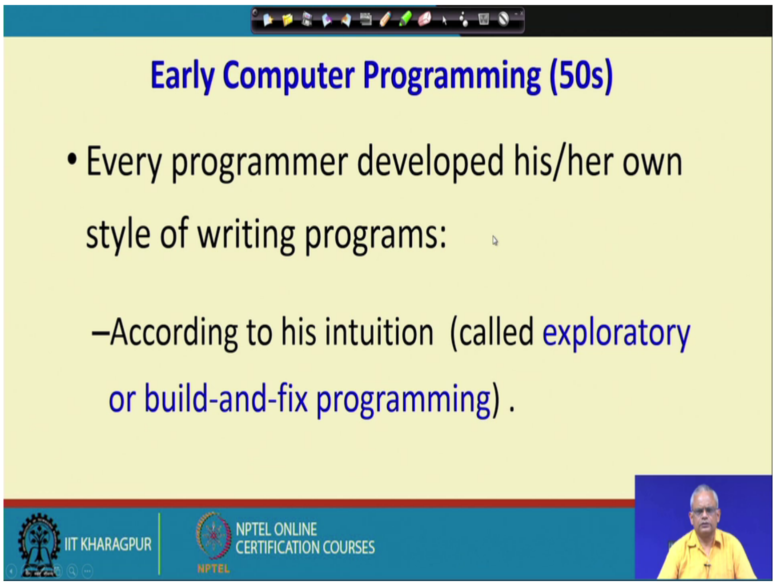
Emergence of Software Engineering Techniques

- Early Computer Programming (1950s):
  - Programs were being written in assembly language...
  - Sizes limited to about a few hundreds of lines of assembly code...

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL

The very early years of computer programming that is 1950s, all programs were written in assembly language and obviously, the program sizes were very small, just few hundreds of lines of assembly code.

(Refer Slide Time: 01:40)

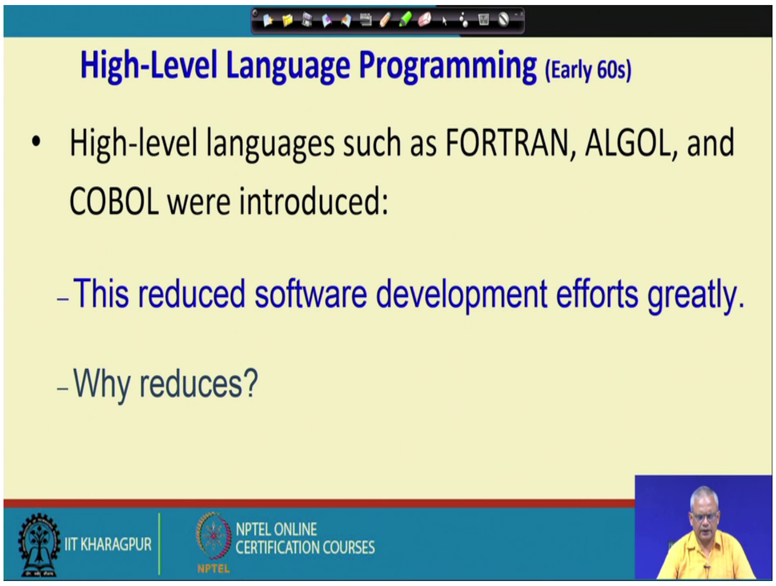


Early Computer Programming (50s)

- Every programmer developed his/her own style of writing programs:
  - According to his intuition (called exploratory or build-and-fix programming) .

And of course, there were no methodologies available. So, the programmers used the build and fix or the exploratory style. Basically, they wrote the program as they thought it right.

(Refer Slide Time: 02:02)



High-Level Language Programming (Early 60s)

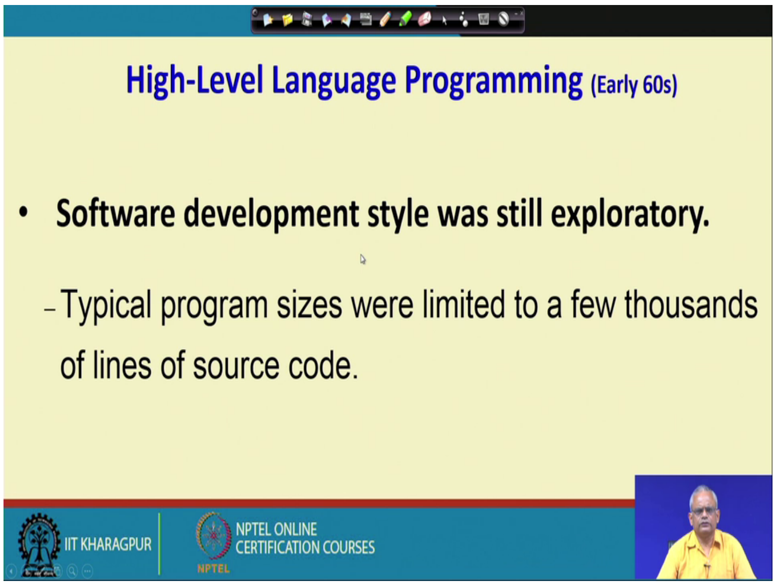
- High-level languages such as FORTRAN, ALGOL, and COBOL were introduced:
  - This reduced software development efforts greatly.
  - Why reduces?

In the next decade a 1960s higher level languages came into being FORTRAN, ALGOL, COBOL, etcetera; and this increase the productivity greatly.

But then, why is it that writing in a high level language is much more productive than writing in assembly language.

There are two reasons here: one is that, in assembly language one would have to write the program considering the register contents, the machines architecture and so on. Using a high level language one writes the program in terms of the variables and so on which corresponds to the real problem or in other words the high level language abstracts the machine details. The programmer need not be concerned about the exact architecture of the machine. And the second reason is that: each high level construct is equivalent to writing 3-4 assembly instructions. And possibly you can say third reason is that the high level language programs are much nearer to the human's language than the assembly language and therefore, it becomes easy to write programs.

(Refer Slide Time: 03:57)

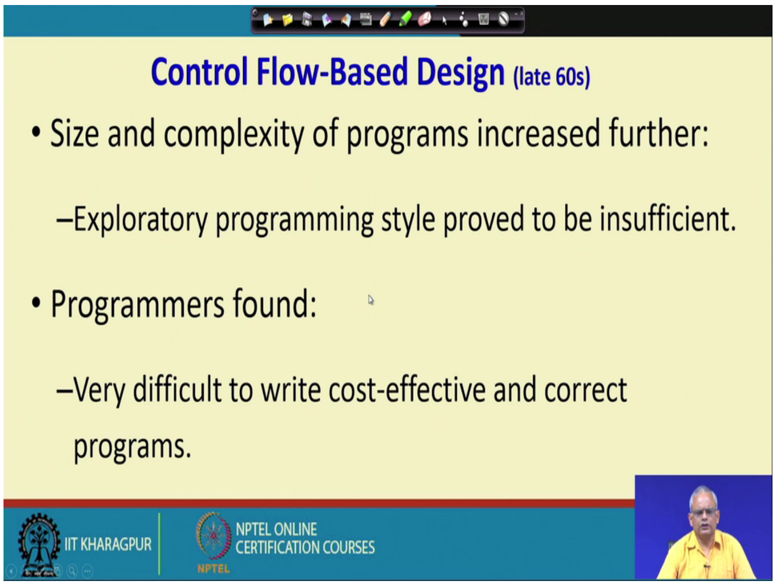


High-Level Language Programming (Early 60s)

- Software development style was still exploratory.
  - Typical program sizes were limited to a few thousands of lines of source code.

But then, even though programs were written in high level language in 1960s still the development was exploratory. Everybody use their own intuition how to write the program. And this worked, because the program sizes were small; just a few thousand lines of code.

(Refer Slide Time: 04:24)



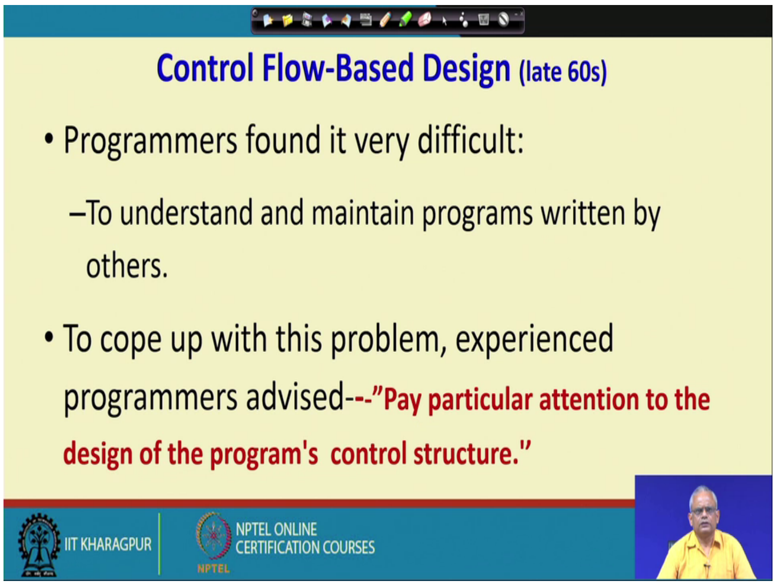
Control Flow-Based Design (late 60s)

- Size and complexity of programs increased further:
  - –Exploratory programming style proved to be insufficient.
- Programmers found:
  - –Very difficult to write cost-effective and correct programs.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

After some time that is towards the end of that decade the program sizes started to increase; larger programs are required. And it becomes difficult for programmers that took long time to write using the exploratory style and also there were lot of bugs and so on.

(Refer Slide Time: 05:00)



Control Flow-Based Design (late 60s)

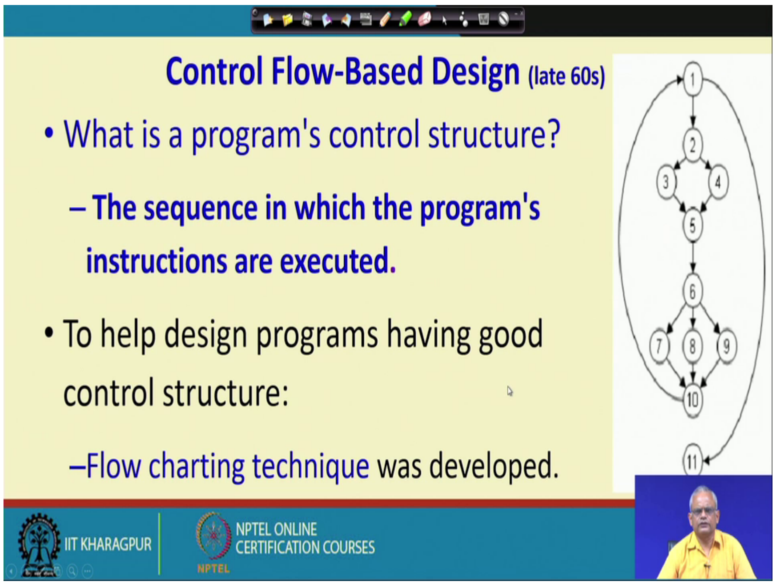
- Programmers found it very difficult:
  - –To understand and maintain programs written by others.
- To cope up with this problem, experienced programmers advised--"Pay particular attention to the design of the program's control structure."

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

But there were good programmers, who were able to write programs satisfactorily and they advised: "pay attention to the design of the program's control structure".

The programmers have to design programs that will have good control structure, but then how do they do it.

(Refer Slide Time: 05:31)



Control Flow-Based Design (late 60s)

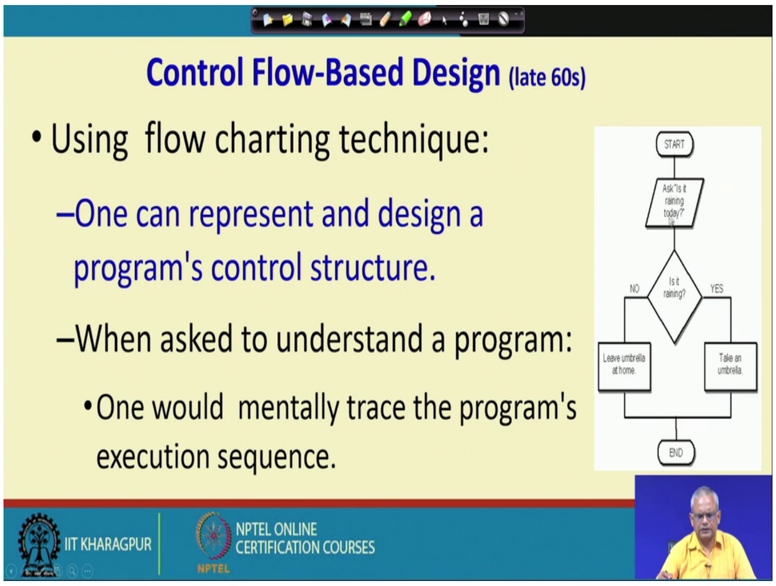
- What is a program's control structure?
  - **The sequence in which the program's instructions are executed.**
- To help design programs having good control structure:
  - Flow charting technique was developed.

1
2
3
4
5
6
7
8
9
10
11
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

Before that let us look at what exactly is meant by a control structure of a program. A control structure of a program is the sequence in which the programs statements can execute. So maybe, after 1 2 will execute statement 2 and 2 is a decision, so 3 or 4 may execute either 3 or 4 and then 5 and so on. So, this gives the program control structure.

But then, how does one go about designing good control structure and for that, the flow charting technique was developed. The idea was that before writing the program, represent the logic or the algorithm of the program in a flow chart and then translate the flow chart into code.

(Refer Slide Time: 06:38)

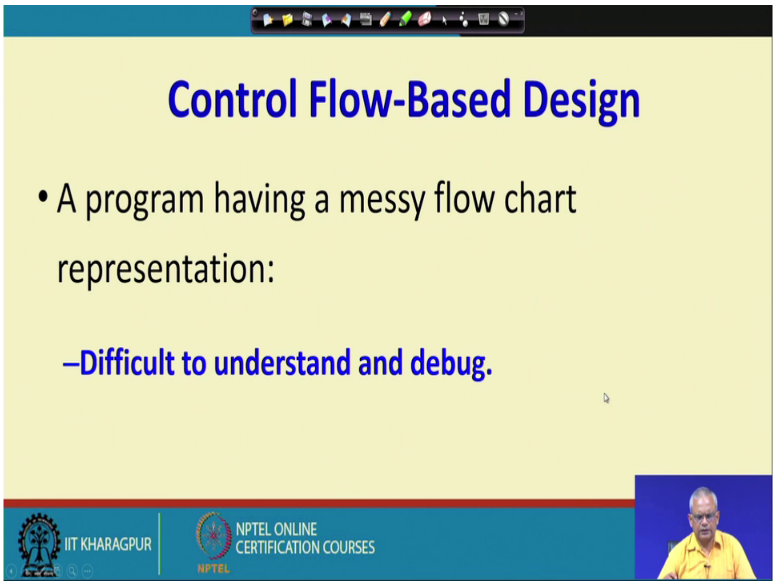


Control Flow-Based Design (late 60s)

- Using flow charting technique:
  - One can represent and design a program's control structure.
  - When asked to understand a program:
    - One would mentally trace the program's execution sequence.

A flow chart like this, and then translate it into code and then it was a good advice and programmers found that it really helped to write good programs; the flow charting technique was extensively used. But then, why is it that a flow charting technique or a good control structure how why is it that this help in writing good quality programs; cost effectively you can write larger programs using control flow based design.

(Refer Slide Time: 07:22)

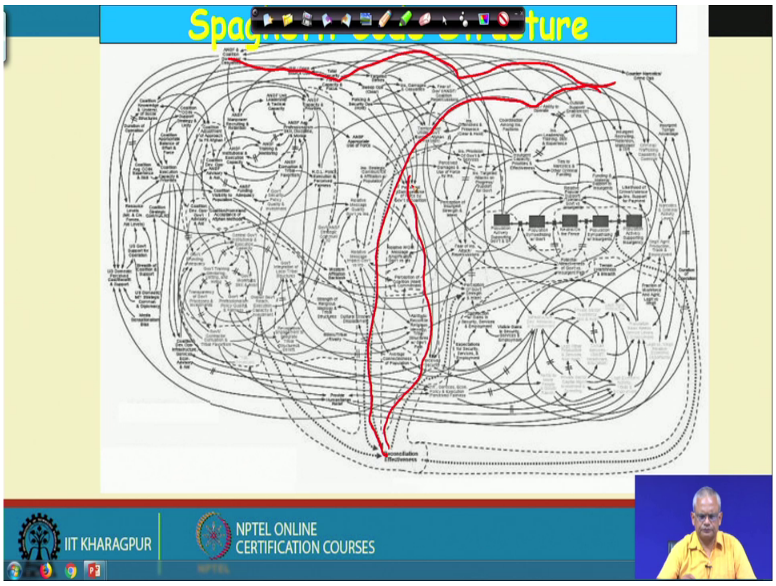


Control Flow-Based Design

- A program having a messy flow chart representation:
  - Difficult to understand and debug.

The answer is that: if we do not use a control structure design and write the program as it comes to mind then the control structure will be poor, and if the control structure will be poor then it becomes very difficult to understand the program and debug.

(Refer Slide Time: 07:44)

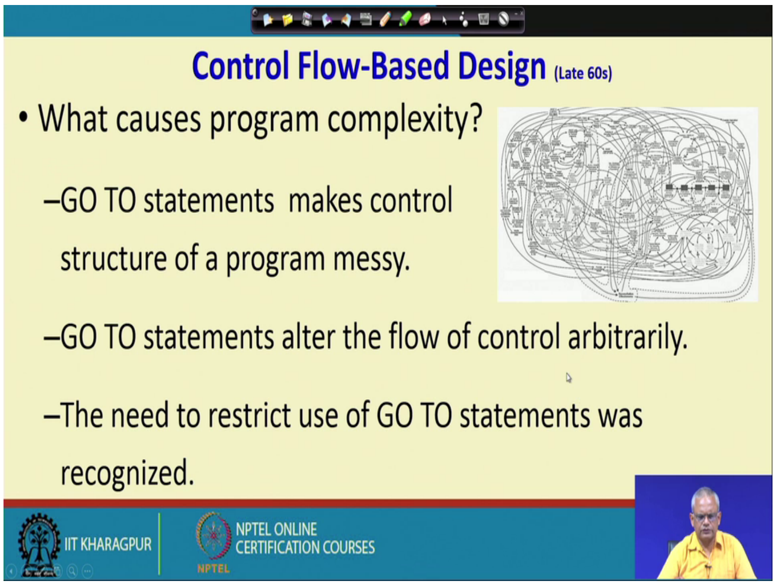


Why is that? Let us look at a program having a bad control structure. To understand a program whose control structure is like this; one would have to find out from every output that is produced, what are the instructions that are executed, sequence of ins that is the sequence of instructions gives that path through the program. That is extremely difficult to trace from where the input might have come.

Not only that, we do not know how many paths are there may be 1000-2000 and for a programmer to understand a program of this control structure you would have to mentally trace all the program paths and it becomes extremely difficult. To understand this piece of code even if you try for a year, 2 year still would not have a full understanding of the program.

But, if you design the control structure using a flow chart: the number of path there will be limited and in reasonable a small time can be able to understand the program debug it and the program development becomes fast the productivity increase.

(Refer Slide Time: 09:37)



Control Flow-Based Design (Late 60s)

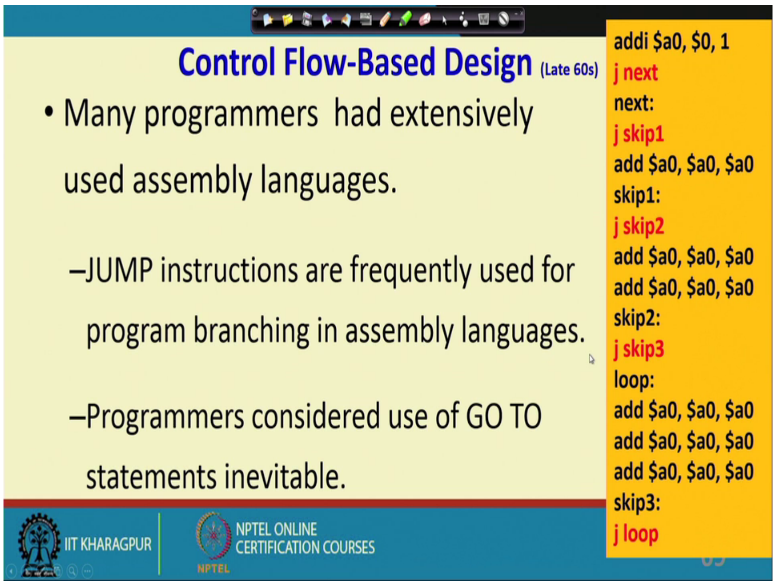
- What causes program complexity?
  - GO TO statements makes control structure of a program messy.
  - GO TO statements alter the flow of control arbitrarily.
  - The need to restrict use of GO TO statements was recognized.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

Now, let us see the control flow base design, it reduce the con complexity. But then, one thing is that besides the flow chart; what are the other things that one must do. One thing that was observed by the experienced programmers is that GO TO statements, they make the control structure bad. As long as a programmer writes program full of GO TO statements then even if you had designed the control structure etcetera, still you will end up writing bad code.

So, the good programmers advised do not use GO TO statements they will ultimately make the program structure bad.

(Refer Slide Time: 10:29)



Control Flow-Based Design (Late 60s)

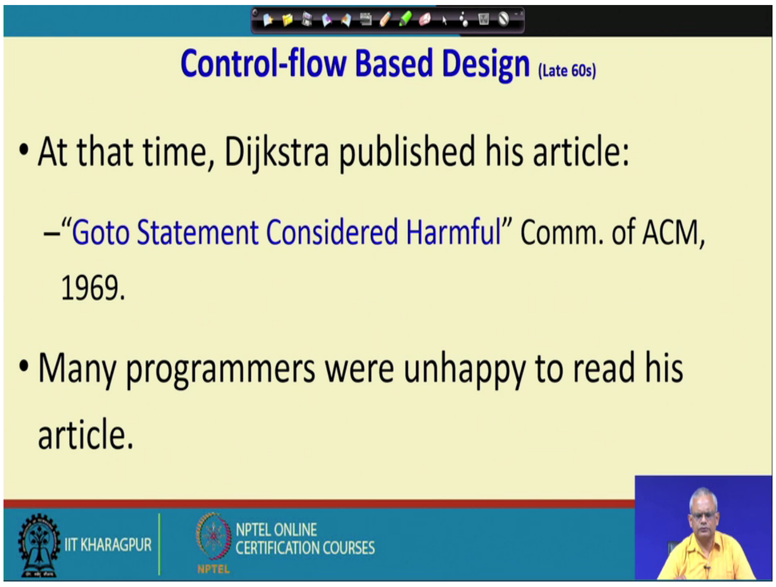
- Many programmers had extensively used assembly languages.
  - JUMP instructions are frequently used for program branching in assembly languages.
  - Programmers considered use of GO TO statements inevitable.

```
addi $a0, $0, 1
j next
next:
j skip1
add $a0, $a0, $a0
skip1:
j skip2
add $a0, $a0, $a0
add $a0, $a0, $a0
skip2:
j skip3
loop:
add $a0, $a0, $a0
add $a0, $a0, $a0
add $a0, $a0, $a0
skip3:
j loop
```

But, that was the period late 1960s and there were many assembly programmers and assembly programmers still being written to a large extent. And any assembly program if you look at it. See, here that there are many jump statements; every assembly program typically has many jump statements and the assembly programmers thought that it would be impossible to write a program without using jump statements. Maybe somebody can reduce the number of jump statements, but how can jump statements be totally not used.

And they wrote many articles actually. They said it is not possible to write program using jump statements or the GO TO statements; the GO TO statements are inevitable.

(Refer Slide Time: 11:41)



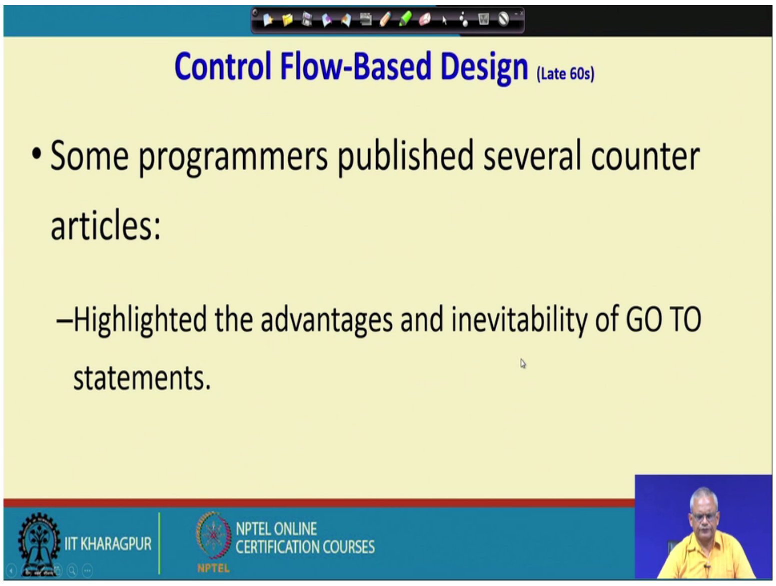
Control-flow Based Design (Late 60s)

- At that time, Dijkstra published his article:
  - "Goto Statement Considered Harmful" Comm. of ACM, 1969.
- Many programmers were unhappy to read his article.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

At that time one of the pioneer in this are Dijkstra, he published a article in the communications of ACM 1969 "Goto Statement Considered Harmful" and he argued in the article that the GO TO statements will not be used unless it is absolutely necessary. Obviously, those who had assembly program in background they were very unhappy to read the article and they wrote many counter articles.

(Refer Slide Time: 12:19)



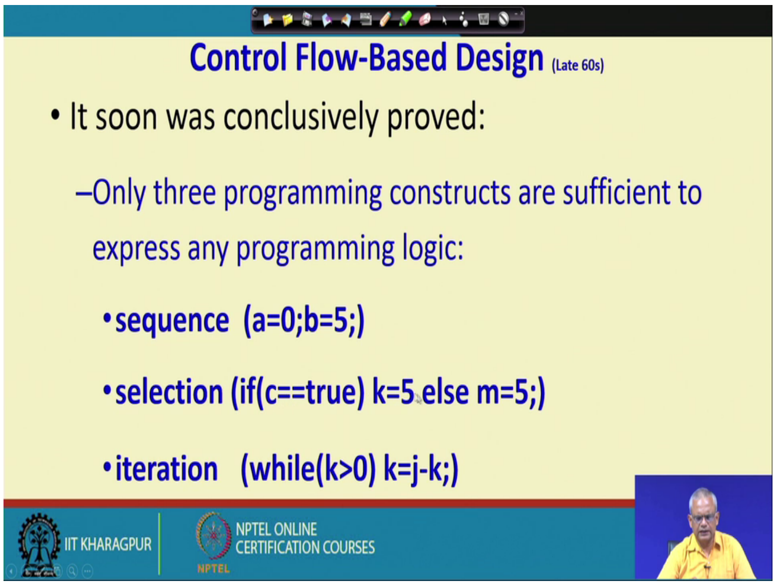
Control Flow-Based Design (Late 60s)

- Some programmers published several counter articles:
  - Highlighted the advantages and inevitability of GO TO statements.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

They said: GO TO statements are actually advantageous they help write efficient programs it is inevitable, you cannot just write program without GO TO statements.

(Refer Slide Time: 12:32)



Control Flow-Based Design (Late 60s)

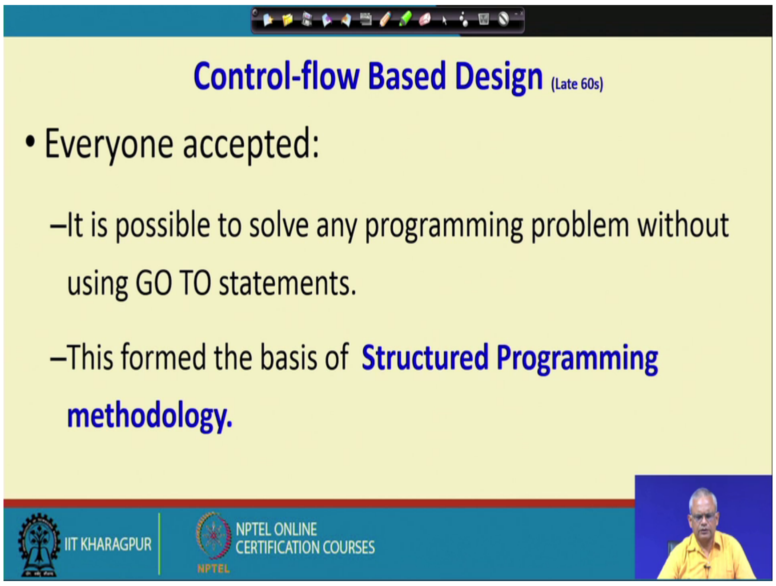
- It soon was conclusively proved:
  - Only three programming constructs are sufficient to express any programming logic:
    - **sequence (a=0;b=5;)**
    - **selection (if(c==true) k=5 else m=5;)**
    - **iteration (while(k>0) k=j-k;)**

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

But in the subsequent year it was proved that to solve any programming problem just three programming constructs are sufficient. To express any programming logic only three types of programming constructs are needed. These are sequence type statements: for example, a statement followed by a equal to 0 followed by b equal to 5 selection statement like if then else switch and so on. And the iteration type of statement for loop, while loop and so on.

And naturally every programming language provides these three categories of constructs; the sequence type of constructs, selection type of constructs and the iteration type of constructs and just see here that there is no place for GO TO.

(Refer Slide Time: 13:43)



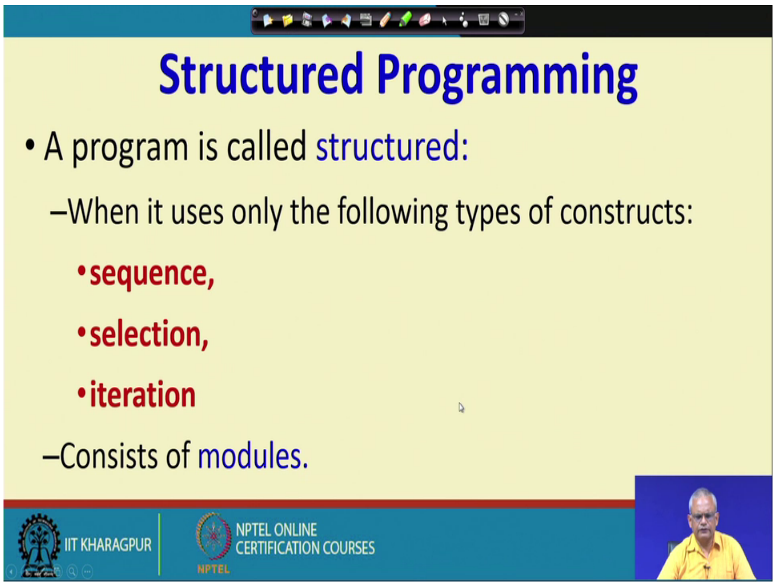
Control-flow Based Design (Late 60s)

- Everyone accepted:
  - It is possible to solve any programming problem without using GO TO statements.
  - This formed the basis of **Structured Programming methodology.**

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

But has GO TO been totally withdrawn away with let us look at that issue, but then a program written without using GO TO statements form the basis of structured programming. That is one of the main requirement of a structured programming.

(Refer Slide Time: 14:05)



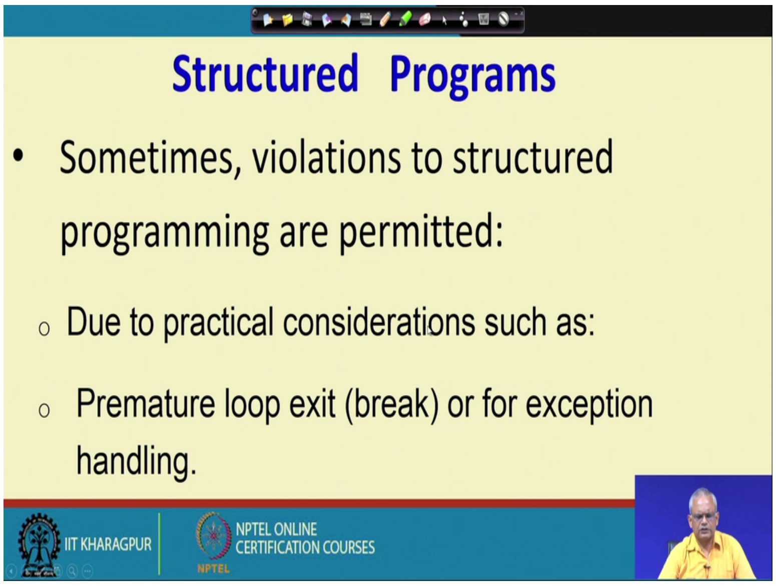
Structured Programming

- A program is called structured:
  - When it uses only the following types of constructs:
    - **sequence,**
    - **selection,**
    - **iteration**
  - Consists of modules.

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

The concept of structured programming no GO TOs are used, only three types of construct sequence, selection and iteration are used. And also a program, specially large programs should be divided into modules but has the GO TO statements been totally done away with.

(Refer Slide Time: 14:33)



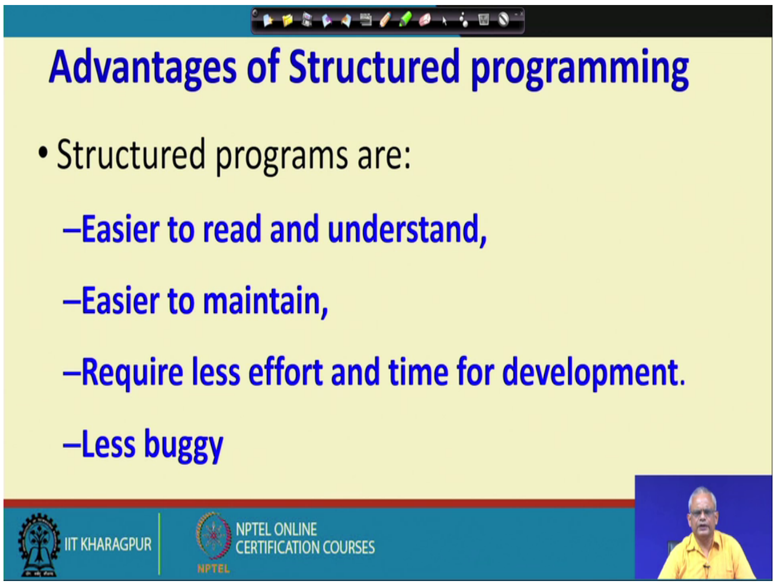
Structured Programs

- Sometimes, violations to structured programming are permitted:
  - Due to practical considerations such as:
  - Premature loop exit (break) or for exception handling.

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

No sometimes they are necessary for practical consideration. For example, premature loop exist like a break statement, so that is actually a form of GO TO statement or let us say exception handling. But then, these are very rarely used and it is not like GO TO statement that is used all through the program these are under very rare circumstances that one would use break and exception handling and so on and these are the characteristics of the structured programs.

(Refer Slide Time: 15:16)



Advantages of Structured programming

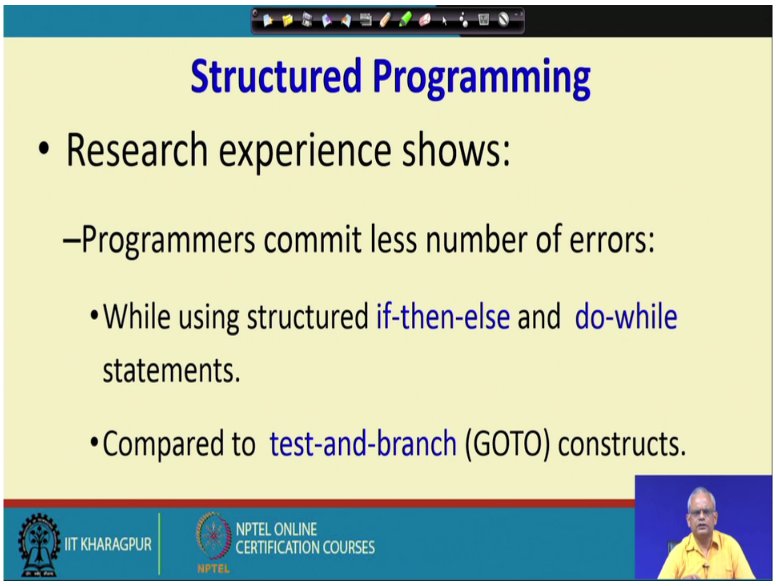
- Structured programs are:
  - –Easier to read and understand,
  - –Easier to maintain,
  - –Require less effort and time for development.
  - –Less buggy

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

But then we must be clear that what advantages does structured program provide, or in other words if one does not write structured programs that is writes using GO TOs it is not modular then what problem you will face. The problem that he will face of course is that the control structure will become bad, it will be difficult to understand and when it becomes difficult to understand the program even the programmer who has written it will find it hard to maintain or any other person trying to maintain it find it extremely harder. And also, if it is not structured to develop the program itself will require too much of effort and time and there will be too many bugs.

Naturally, structured program has lot of advantages and it was advised that write structured programs only; that is using the sequence, selection, iteration constructs and modular programs. Even though, the language used is supported GO TO statements and so on the use should be very restricted. Of course, now the programming languages do not support GO TO statements and therefore, a programmer by default writes structured programs and these programs are modular and so on. So, these are structured program and it has become inbuilt into the programming languages.

(Refer Slide Time: 17:16)



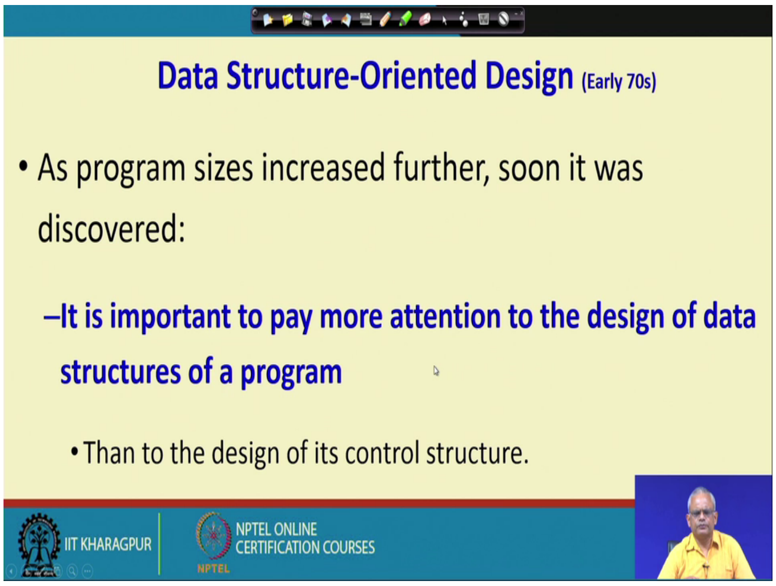
Structured Programming

- Research experience shows:
  - –Programmers commit less number of errors:
    - While using structured if-then-else and do-while statements.
    - Compared to test-and-branch (GOTO) constructs.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL

And this was proved the advantages of structured programming was proved through large number of experiments observation.

(Refer Slide Time: 17:32)



Data Structure-Oriented Design (Early 70s)

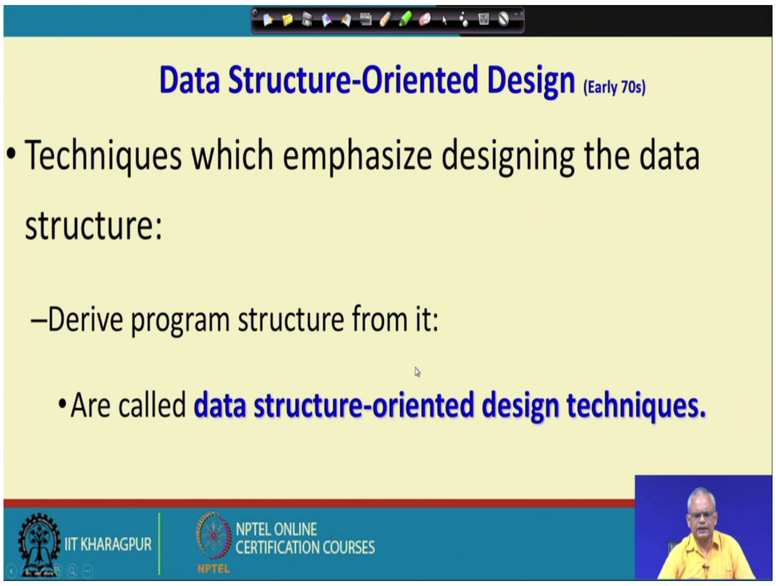
- As program sizes increased further, soon it was discovered:
  - –It is important to pay more attention to the design of data structures of a program
    - Than to the design of its control structure.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

But as the program size has kept on increasing further 50000 lines of code and so on, just paying attention to control structure was not really sufficient. Still the programming was becoming very expensive, develop large program usually takes long time, full of bugs, costly and so on.

At that time it was realized that; yes, control structure is important to pay attention to, but what is more important for large programs is to pay attention to what kind of data structure we use. The data structures should be designed very carefully.

(Refer Slide Time: 18:30)



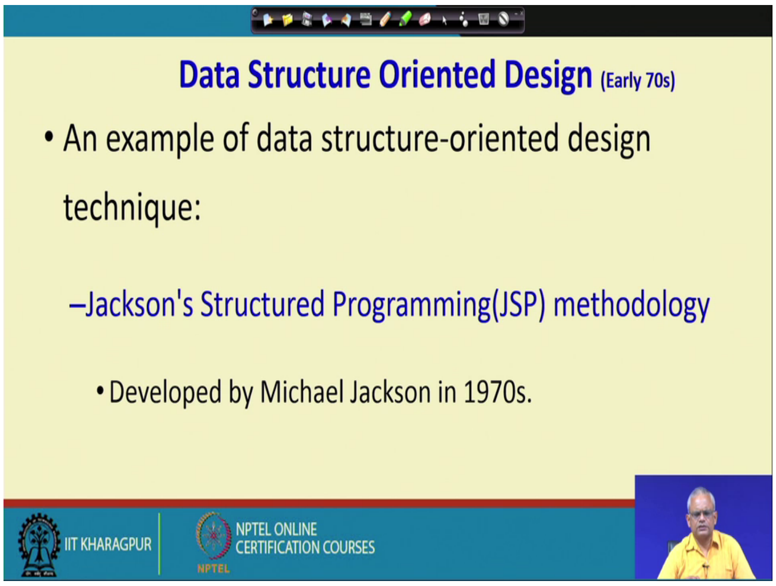
Data Structure-Oriented Design (Early 70s)

- Techniques which emphasize designing the data structure:
  - –Derive program structure from it:
    - Are called **data structure-oriented design techniques.**

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

And these are the data structure oriented designed in early 1970s. Here, the philosophy or the main idea here is that first design the programs data structure. As long as suitable data structure has been designed then the program structure can be derived from it.

(Refer Slide Time: 19:01)



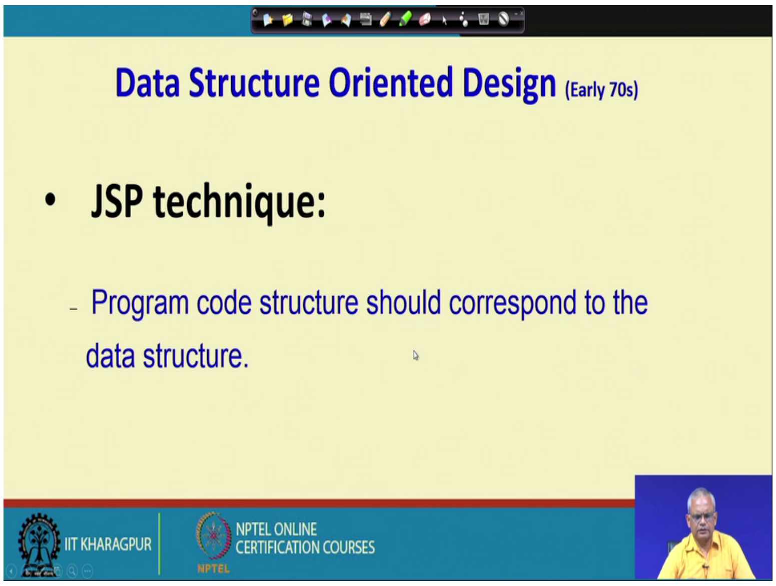
Data Structure Oriented Design (Early 70s)

- An example of data structure-oriented design technique:
  - Jackson's Structured Programming(JSP) methodology
    - Developed by Michael Jackson in 1970s.

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

Many data structure oriented design techniques became popular in 1970s, the ones that stood out the Jackson's Structured Programming or the JSP methodology.

(Refer Slide Time: 19:17)



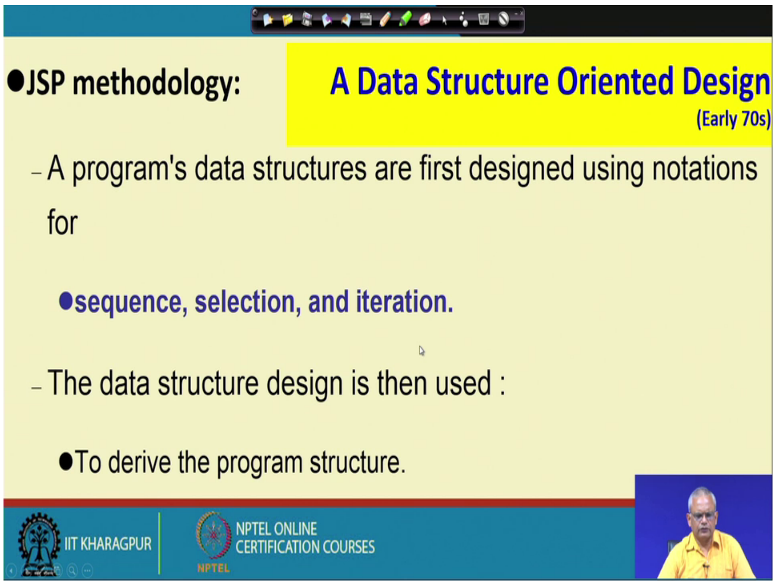
Data Structure Oriented Design (Early 70s)

- **JSP technique:**
  - Program code structure should correspond to the data structure.

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

So, there the provided constructs of how to design the data structure first, how to represent the data structure in diagrammatic form and then how to write a program based on the data structure.

(Refer Slide Time: 19:39)



●JSP methodology: A Data Structure Oriented Design (Early 70s)

– A program's data structures are first designed using notations for

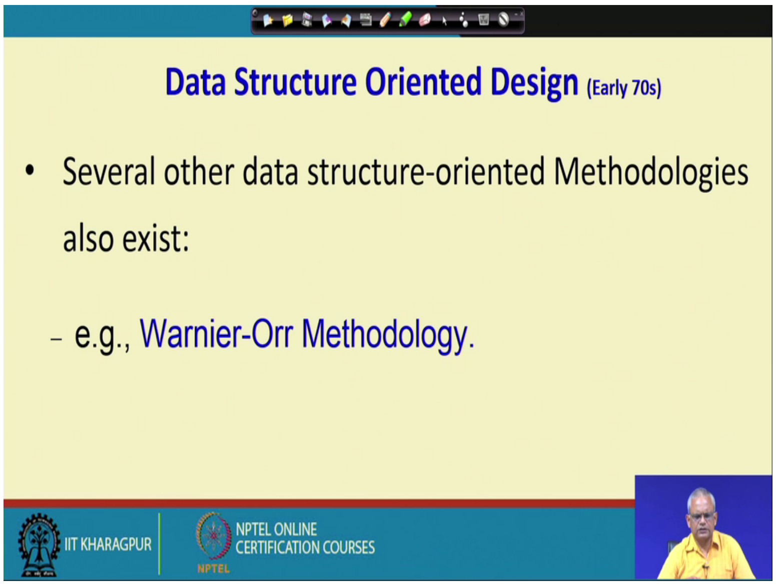
●sequence, selection, and iteration.

– The data structure design is then used :

●To derive the program structure.

So, they had notations for representing sequence, selection, and iteration. The data structure designed using the notations for sequence, selection and iteration and then these are translated into program structure.

(Refer Slide Time: 20:00)

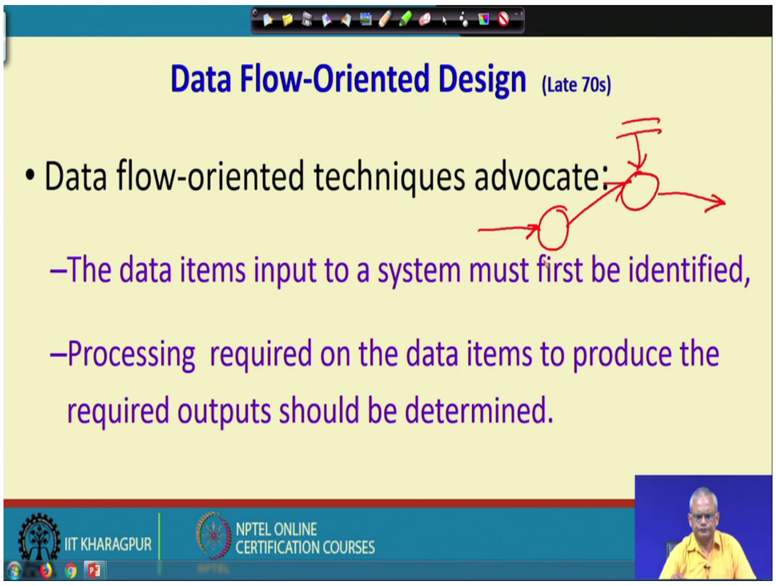


Data Structure Oriented Design (Early 70s)

- Several other data structure-oriented Methodologies also exist:
  - e.g., Warnier-Orr Methodology.

There were other data structure oriented design technique; namely the Warnier-Orr Methodology was also very popular.

(Refer Slide Time: 20:11)



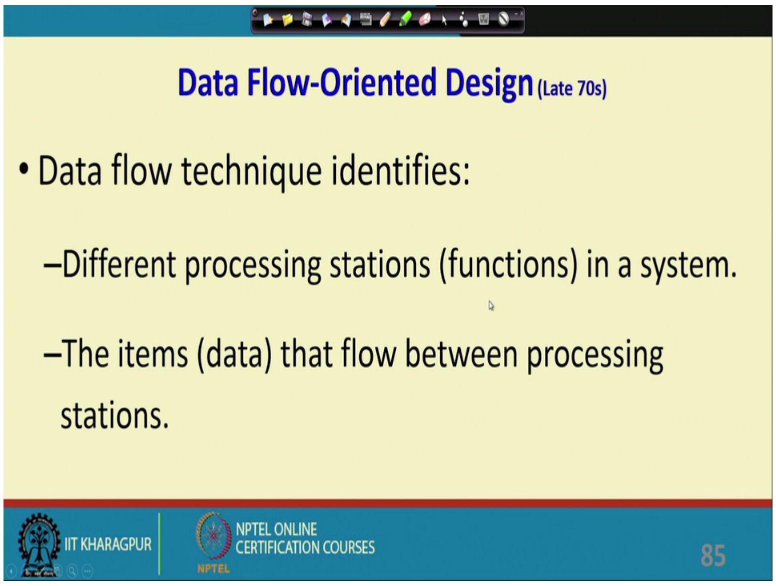
Data Flow-Oriented Design (Late 70s)

- Data flow-oriented techniques advocate:
  - –The data items input to a system must first be identified,
  - –Processing required on the data items to produce the required outputs should be determined.

But then as the program I just still further increase with, the data flow oriented design techniques came into picture. Here the main idea is that to take control structure data structure these are important. But, to get really good program we must pay particular attention how the data flows through the program. That is starting from the input what kind of processing is done until the output is done. It had a notation that is a DFD- the data flow diagram notation where we can first represent the data flow it is from the input what are the processing done until.

So, we will have the data coming in and we will have processing done, maybe some data stored data will be used until the output comes out. And once we have the data flow representation for a program it can be transformed into the design and that gives good design.

(Refer Slide Time: 21:53)



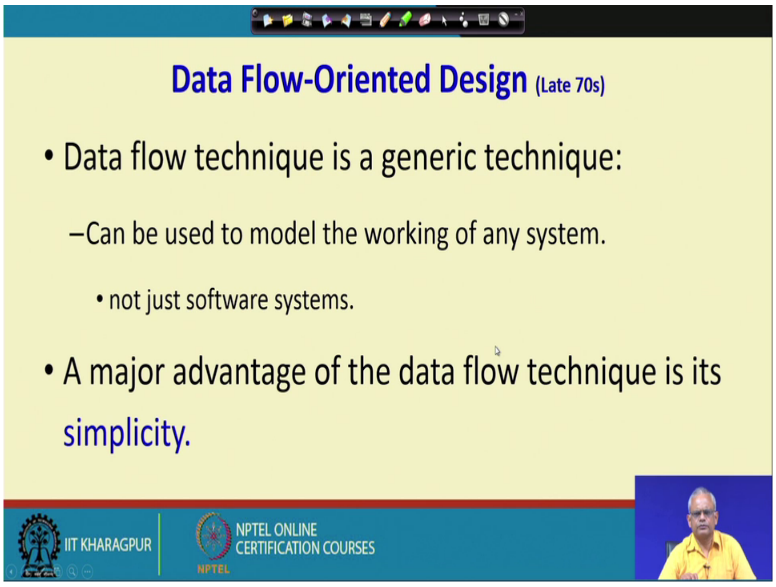
Data Flow-Oriented Design (Late 70s)

- Data flow technique identifies:
  - Different processing stations (functions) in a system.
  - The items (data) that flow between processing stations.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 85

Here, starting with the data input the processing that is done are called as the processing stations or functions, and how the data flows between these processing stations is presented and from here the design is obtained.

(Refer Slide Time: 22:16)



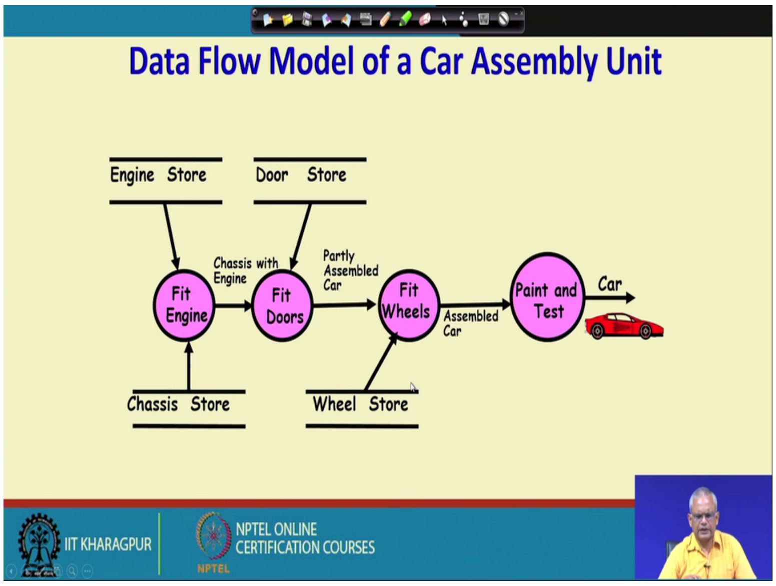
Data Flow-Oriented Design (Late 70s)

- Data flow technique is a generic technique:
  - Can be used to model the working of any system.
    - not just software systems.
- A major advantage of the data flow technique is its simplicity.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

The data flow diagram is a very simple techniques, it takes hardly an hour or so to learn this technique; very simple technique. And the technique became very popular, can be used to model the working of any system not the software system. And as you are saying that it is extremely simple; as during the course of our lecture we will just spend small amount of rime to draw the DFD representation of a problem.

(Refer Slide Time: 22:56)

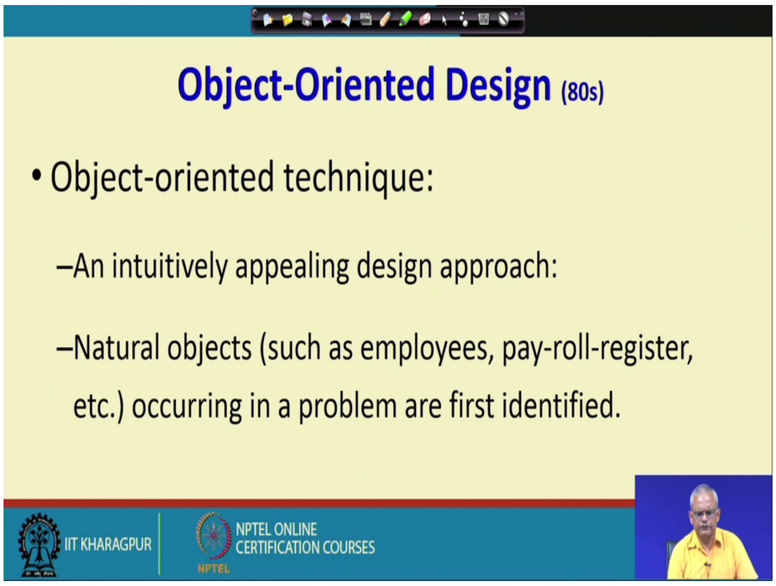


Data Flow Model of a Car Assembly Unit

Let us say how simple it is.

Let us say we have a car assembly unit, where there are processing stations. One processing station takes engines and chassis and it fix the engine to the chassis. And then the chassis with the engine goes to another processing station, where it takes out doors from the door store fits door to the chassis. And then this, proceeds to the next processing station where it fits wheels gets wheels from the wheel store. This is the representation two parallel lines which store, gets it fits the wheels and then the assembled car comes out and is painted and is ready to be dispatched.

(Refer Slide Time: 23:54)



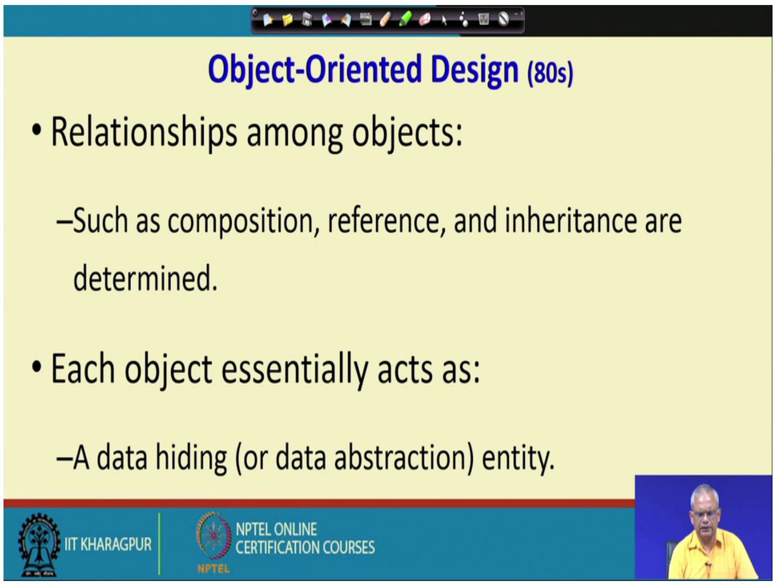
Object-Oriented Design (80s)

- Object-oriented technique:
  - –An intuitively appealing design approach:
  - –Natural objects (such as employees, pay-roll-register, etc.) occurring in a problem are first identified.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

And subsequently the object oriented technique in 1980s came into picture. This had the appeal of representing the natural objects in the problem. As they occur in the problem, like employees pay roll, register, and so on. And then, writing the program using these objects will look at the object oriented design technique it has become quite sophisticated now, as we proceed we will look at that.

(Refer Slide Time: 24:50)



Object-Oriented Design (80s)

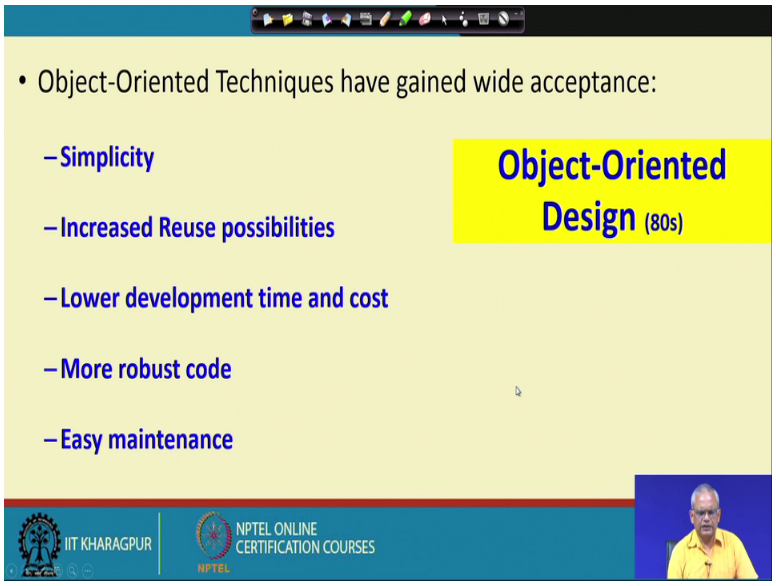
- Relationships among objects:
  - –Such as composition, reference, and inheritance are determined.
- Each object essentially acts as:
  - –A data hiding (or data abstraction) entity.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Once we identify the objects, represent the relations among the objects like composition reference, inheritance and so on.

One of the main advantage of the object oriented design is that it leads to a very good modular design, because the objects are essentially similar to a good modules which have data hiding or data obstruction entity as we proceed these terms will become clear. What we mean by data obstruction or data hiding entity and why is this a good idea.

(Refer Slide Time: 25:33)



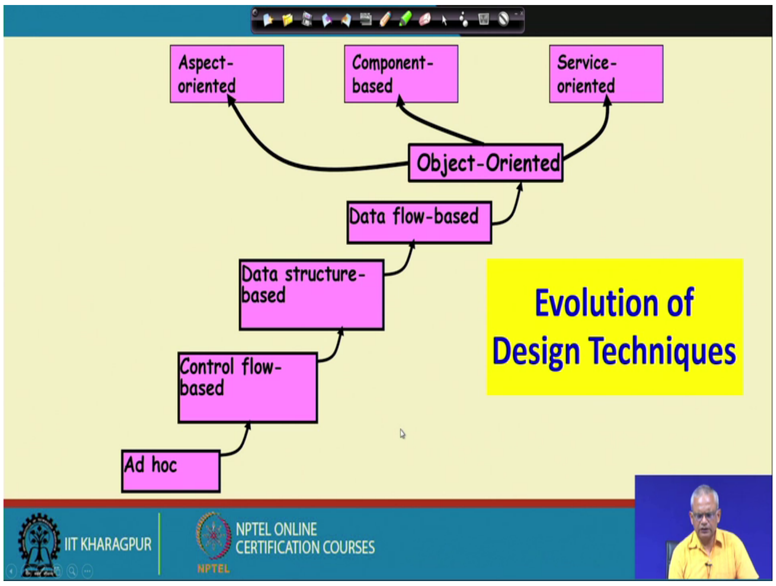
- Object-Oriented Techniques have gained wide acceptance:
  - Simplicity
  - Increased Reuse possibilities
  - Lower development time and cost
  - More robust code
  - Easy maintenance

Object-Oriented Design (80s)

Many advantages of the object oriented design technique. The design is simple just by looking at the design of a large and complex problem we will be able to in very short time have a fair idea about how the program will work. It helps reuse, it lowers development time, development, cost produces code which is less buggy easy to maintain.

The object oriented design technique has many advantages and no wonder that now this technique is being used extensively.

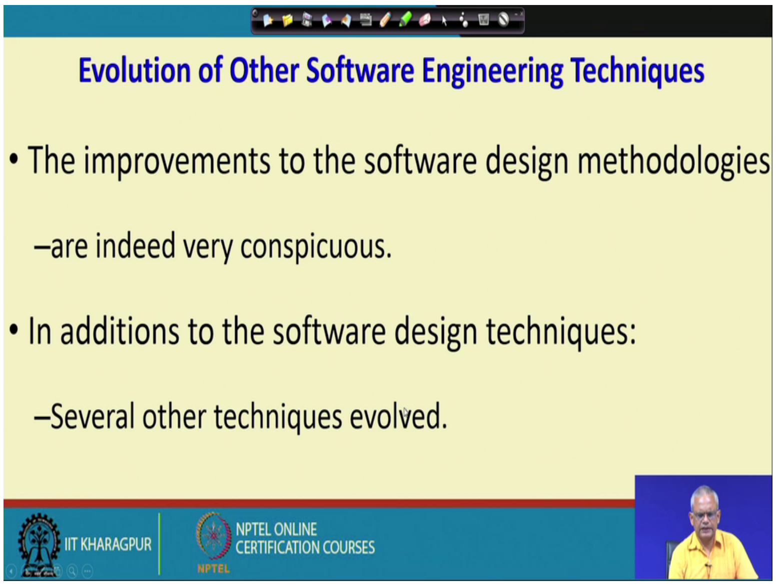
(Refer Slide Time: 26:22)



If we look at the evolution of the design techniques, see that starting from the Ad hoc, the control flow based, data structure based, data flow, object oriented, and then later the aspect oriented, component based, service oriented and so on.

So, the development in this area has been rapid compared to the programs that were written in 1950s and so on, right now we use very sophisticated techniques to write the program and as we proceed through this course we will examine those techniques.

(Refer Slide Time: 27:08)



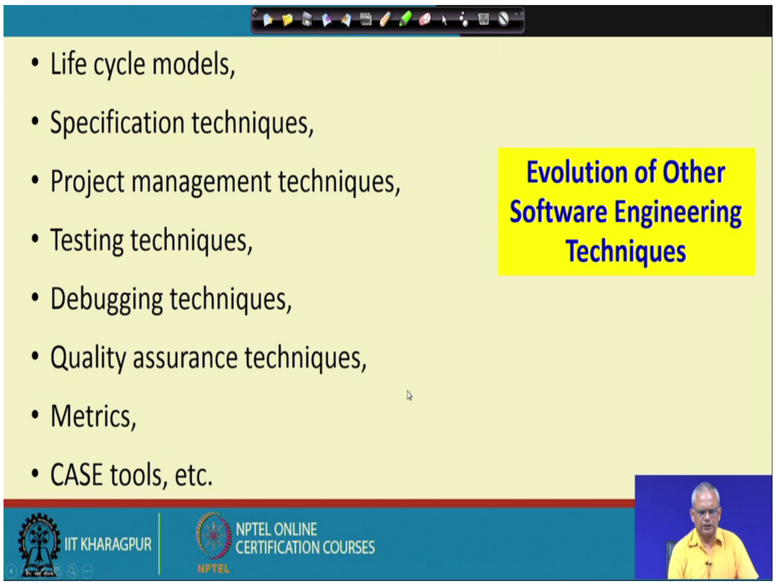
Evolution of Other Software Engineering Techniques

- The improvements to the software design methodologies
  - –are indeed very conspicuous.
- In additions to the software design techniques:
  - –Several other techniques evolved.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

Not only the design techniques, the other things other software engineering principles are also rapidly evolved.

(Refer Slide Time: 27:22)



- Life cycle models,
- Specification techniques,
- Project management techniques,
- Testing techniques,
- Debugging techniques,
- Quality assurance techniques,
- Metrics,
- CASE tools, etc.

Evolution of Other Software Engineering Techniques

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL

For example: we have the life cycle models, earlier no life cycle models the programmer just wrote the code. We have good specification techniques which will examine project management techniques, testing techniques, which we will examine debug technique, debugging techniques, quality assurance, matrix, case stools, etcetera.

So compared to 1950, slowly various software engineering techniques have evolved year after year and right now, we have a reasonably sophisticated set of techniques which a developer must know to be able to write good programs.

Now the time is over. We will stop here, and we will continue in the next lecture from this point.

Thank you.