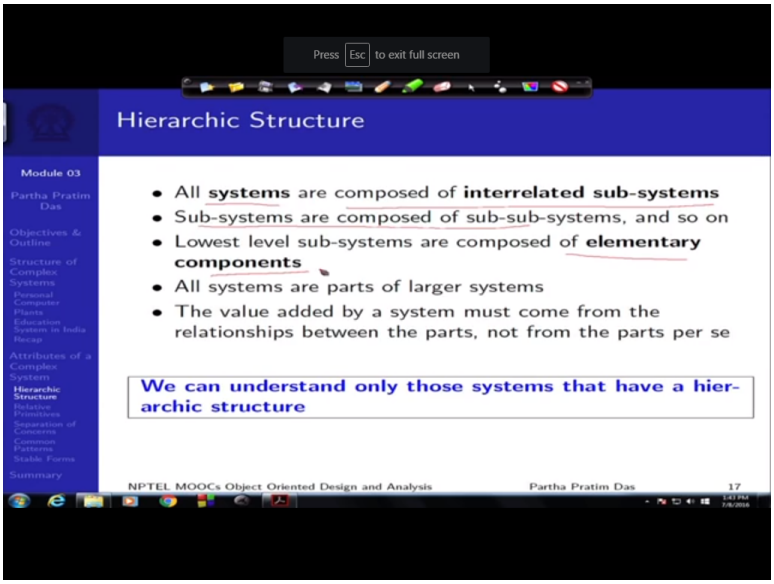
Object-Oriented Analysis and Design
Prof. Partha Pratim Das
Department of Computer Science and Engineering
Indian Institute of Technology – Kharagpur

Lecture – 05
Structure and Attributes of a Complex System (Contd.)

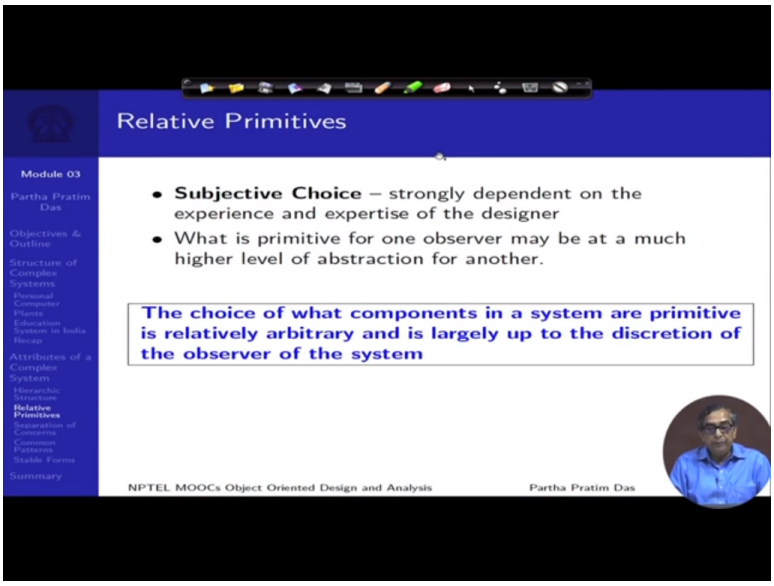
Welcome back to module 3 of object-oriented analysis and design in this module earlier we have looked at three examples of complex systems and tried to characterize their structure and we have come up with some common conclusion leading with the fact that every complex system has to be hierarchic in nature and with that we are now trying to conclude that there are 5 major attributes of complex systems. The hierarchic structure, the relative primitive, separation of concerned, common patterns and stable intermediate forms.

(Refer Slide Time: 00:58)



And we have also taken a brief look into hierarchic structure that is we say that every system is composed of interrelated subsystems and sub-sub systems till you go to an elementary component level. This is a basic property that we say will exist in any complex system that we analyze.

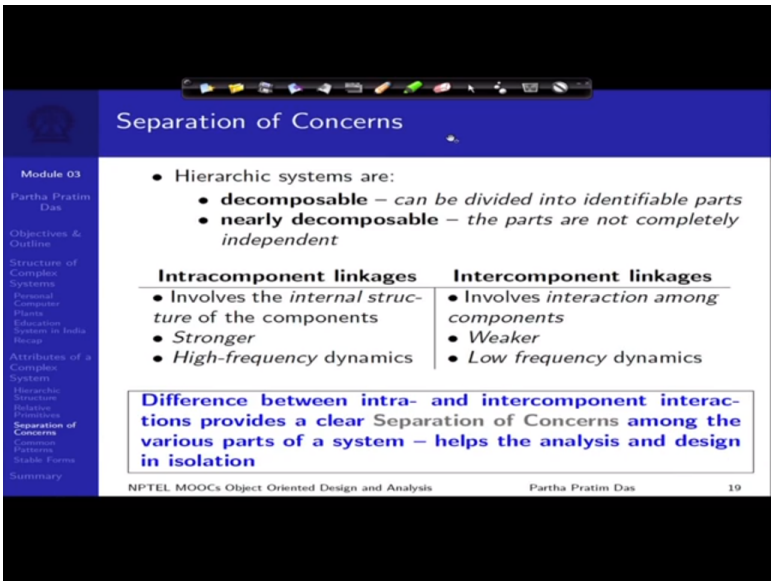
(Refer Slide Time: 01:22)



Now next is relative primitive that is if we keep on decomposing the system in the hierarchy manner as we have said and then there are some where you will have to stop and that stopping point is called the relative primitive or the primitive. It is relative because there is, there cannot be a unique definition of what is a primitive. It is a matter of perception of the observer of the designer as to deciding what is a primitive.

And what could be a primitive for me may not be a primitive for someone else and vice versa. So the choice of what component in a system is primitive is relatively arbitrary and is largely up to the discretion but you will have to for any system if the description has to have some primitives in terms of which things are defined. For example in personal computer, electronic systems the primitive could be taken either as a digital logic gate or as a cmos gate.

Or you can even go lower or you can become even higher and say that no it might primitive it is not up to the level of NAND gate but might primitive for certain electronic subsystems like (()) (02:38) and resistors and I am just doing a rtl kind of design so primitives are relatively arbitrary. (Refer Slide Time: 02:46)



The next is, the next attribute of complex system is separation of concerns that is hierarchic systems are decomposable that is what we have seen then they can be divided into identifiable parts, personal computer decomposable in terms of CPU, hard disk, monitor, keyboard, and so on. At the same time, interestingly they are nearly decomposable in the sense that while we say that these are all decomposable but they are not completely independent.

If CPU is completely independent of the hard disk and completely independent of the monitor and so on then the CPU will not be able to access the file from the hard disk and use it. Monitor would not be able to display the values that the CPU wants to display. So on one side they are decomposable but other side they are nearly decomposable so this kind of can be characterized by the amount of interconnection, inter component linkages that exists.

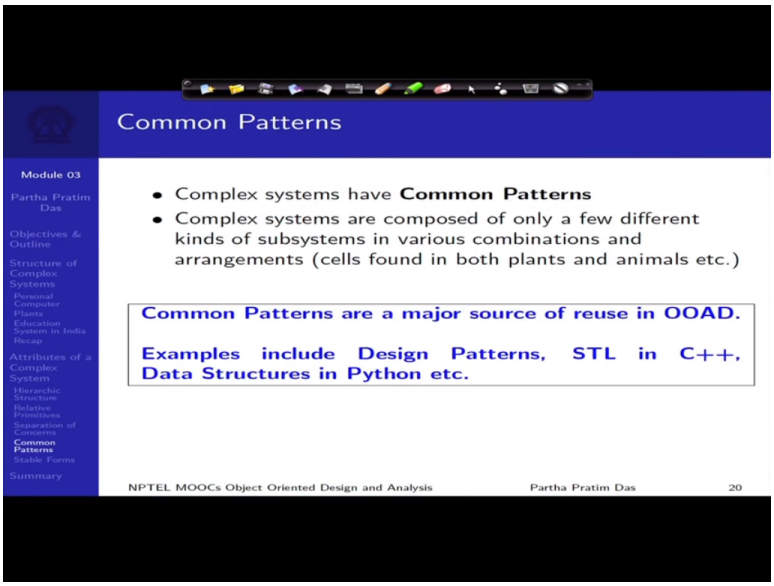
If you look into the CPU there are different components of ALU, primary memory and bus and all that naturally the interconnection that exists between them or inter component linkages that will have to exist between them is much higher compared to what will exist between the CPU and the hard disk. So if we look into this and try to characterize between this intra component linkages. So if you look at this, here we are talking about the intra component that is within component.

And this is inter component that is across component so this involves inter connect interactions between the internal structure and this involves interaction across components then we can always expect that this will be much stronger there will have to be lot more within the system than what happens across the system. This will have a much higher frequency; this will have a much lower frequency.

So the difference between the intra and inter component interaction will provide a clear separation of concern that is when I am looking into the interactions within the CPU that is highly dense and that defines what I understand to be the CPU and when I look at the interaction between the CPU and the memory, it is low frequency, it is weak and that defines that I am talking to a separate system and memory subsystem will have to take care of its own intra component interactions.

But the interaction between the CPU and the memory will be separate and can be separately handled. Now this separation of concern is very very important because that is what allows us to study the subsystems independently. If the separation of concern was not possible then we would not have been able to study, design and develop the subsystems independently. So this third attribute is very critical from our object-oriented analysis design context.

(Refer Slide Time: 05:51)

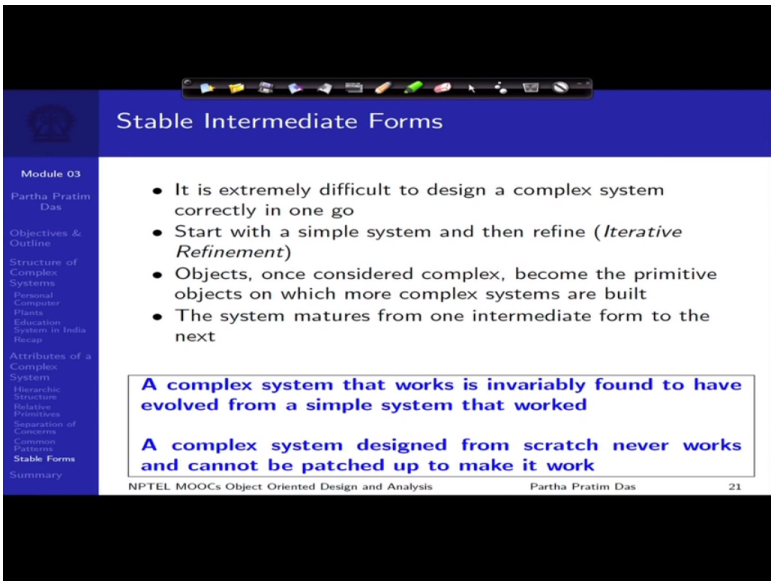


The third that we observe is common pattern we did mention about some of these in terms of saying that we have processors in personal computers we have processors in cards we have processors in mobile phones and so on so. If I understand processor, then I understand all of these. So finding out this commonality is a mee is one of the challenges or one of the major handles through which OOAD can simplify problem.

Because a lot of analysis experience of people show that at the end, not too many different kinds of subsystems amongst many of the common complex systems. So if we can understand those subsystems well over a period of time, then we can approach and analyze several varieties of complex systems quite easily. For example the ehh from the natural world the cells are found both in plants and animals, vessels are found both in plants and animals.

So if one understand cells then in plants then it is much easier for the person to go and understand cells in animals. So common patterns are major source of reuse, so hope you all would be aware with the term of reuse that something which is built can for one purpose can be engaged in another purpose. This reuse becomes easy this reuse becomes possible because there are common patterns.

And if I just talk about very specific examples you have different libraries in the c you have standard libraries in c plus plus you have template library, you have design patterns, you have 5 common data structures in python. These are all examples of common patterns that happen across domains across systems and allow us to make reusable solutions using the ehh object object-oriented analysis and design. So this is fourth attribute which is critical for such systems.
(Refer Slide Time: 07:56)



The 5th and final attribute talks about stable intermediate for. It has to do more with how we should tackle the development of a software system, development of a system which is complex. It is extremely difficult to design a complex system correctly in on go I mean if we just think that we have understood, studied a lot and we can just go and design a complex; we will inevitably face failure.

So what we what succeeds is starting with a simple system and then refine, add some more complexity and then refine again. so this approach is typically called the iterative refinement so the when you start, you say certain objects are very very complex so you start you say okay I am going to make a matlab kind of system and when I start, I need to represent numbers. Now I need to represent number.

That representing a number itself is a very complex concept because you need to deal with bits. The bits are finite in size, they will overflow, underflow then it is high in all that so you are dealing with concepts objects which are just numbers and you find them to be really difficult to handle, they are complex in nature while primitive remains to be the bits that you can manipulate. Once you have built up these objects.

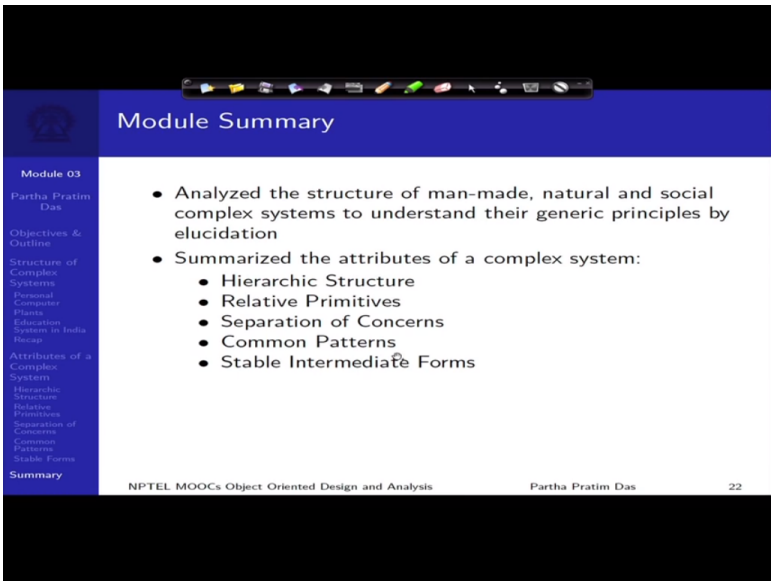
Once you have built up these numbers of say integer, double, floating, floating point numbers, complex numbers, fractional numbers and so on. Then you can use them to build bigger

functionality now you are building up matrices, now you are building up vectors because you are doing a matlab. So now the objects like floating point number which looked very complex and was not considered to be primitive, now in turn it will become primitive.

And now you are making matrix of double numbers, floating point numbers. So now double becomes a primitive and matrix is the next level of object, complex object that you are trying to build up. Once you have been able to build that up, then you say matrix becomes a primitive and I am making a system of linear equation solver where matrix is there, vector is there at every stage. So what is primitive and what is complex is a relative term.

And that is what we take into advantage when we want to, when we need to build up systems and we need to go from one state of the system, one stage of the system to the next one by refinement and this is what is called a intermediate form. So when I have started with representing number or by representing matrices or by representing systems of equations, I have not done my matlab but they are all intermediate forms but what we need is the intermediate form has to be stable by itself.

That is whatever I make of a system to represent and manipulate matrices must be stable to work when I build up more systems on that. So being able to handle stable intermediate forms or having stable intermediate forms is a necessary requirement of complex systems and any attempt to design, try to design complex systems from scratch will inevitably lead to disaster.
(Refer Slide Time: 11:20)



So these are the 5 attributes that is what we have studied so if we sum up, we have analyzed the structure of man-made natural and social kind of, administrative kind of natural system to try to understand the generic principle by elucidation. We are… There is no there is no proven theory that this will happen, things will happen the way we are trying to characterize but this is by illustration by example that we have tried to understand that there is a lot of common properties that exist.

And then we have scan through them, summarize them in terms of 5 basic attributes of a complex systems looking from the object-oriented analysis and design perspective, the hierarchic structure of the system, the relative primitives, separation of concerns, common patterns across different systems and across domain and the iterative refinement leading to stable intermediate forms at every stage of the system.