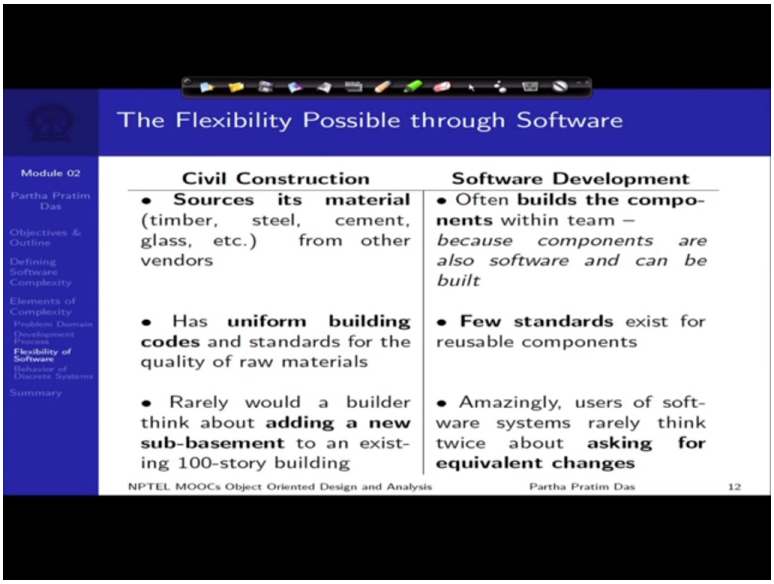
**Object-Oriented Analysis and Design**
**Prof. Partha Pratim Das**
**Department of Computer Science and Engineering**
**Indian Institute of Technology, Kharagpur**

**Lecture – 03**
**Complexity of Software (Contd.)**

Welcome back to module 2 of object-oriented analysis and design. In the earlier part of this module, we have talked about industrial strength software, the software which is complex, large resource constraint, needs a lot of people to develop and so on. We are focusing on the requirements of the development, design, analysis of such software and we have noted that the software is a very complex phenomenon and the complexity comes from 4 different elements of which 2 elements we have discussed.

The complexity that arise from the problem domain including the difficulty of understanding the domain. The difficulty of functional requirements, non functional requirements as well as communication gap between the developer and the user and the external complexity and the other element which we have looked into is the difficulty of managing development teams and we have seen that complex softwares in spite of the use of the all different technology today.

The very high level language reusability, modularity, decomposition. In spite of all of that given a small component of software goes well beyond the comprehension of individuals and therefore it becomes software. Development has become a major activity in team development and team management several teams in several corners of the world often having different issues in interaction between themselves make it a real management challenge to keep unity and uniformity of the design process.
(Refer Slide Time: 02:17)




Now let us look into the third element of software complexity that is flexibility through possible through software. Software is soft it is flexible now the first there are primarily 3 points which you should be sensitive about. The third one you are already aware of sure but the first point is if you just compare software development with civil construction if a building is to be constructed with this components it needs raw materials.

Now no civil construction vendor would start by planting trees so that timber can be used or by setting a steel roll mill so that the steel can be used in casting the concrete and so on. What you will do you will source the raw materials from several vendors and then just will apply your expertise to construct it. In software this is the big problem, the problem is the components of a software or the software themselves the components of the component are in turn again software and so and so.

At every point you are faced with a challenge of whether to build or to buy, whether to build or to borrow, whether to build or to steal. So often it turns out that a team might end up building not only the software that the team needs to build but it also builds a whole lot of components second point which you should understand is any field of engineering like civil construction, when the source are raw material there are several uniform building codes and standards which control the quality of the law material.

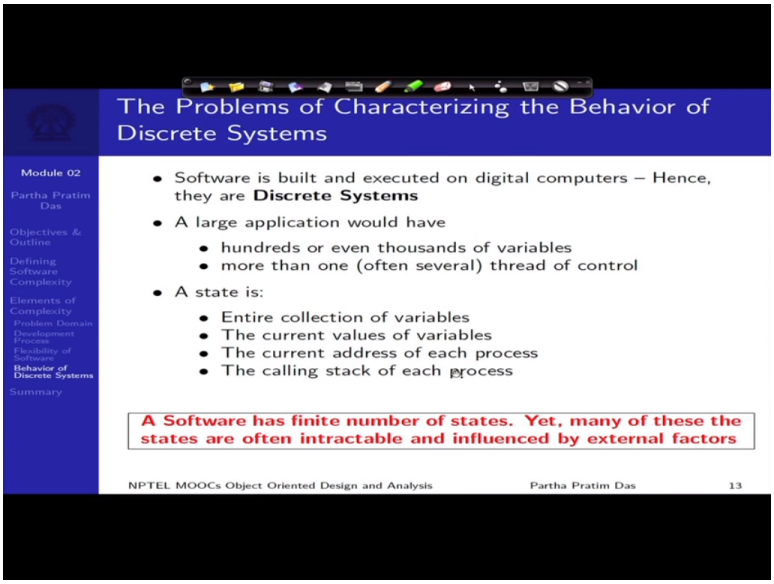
So if you have to cast a cast a concrete the kind of grain that the sand should have the kind of casting strength you cement should have are all documented and fixed unless you adhere to those you cannot construct and therefore when you take raw materials you will always check against those. in software

unfortunately there are very few standards really really hardly any standard to check if a component is really usable what is the reliability of a component that you are using and so on.

So the dimensions of flexibility at the major issue of component development become a part of the software development. The third point which all of us know is if you consider you have a 100 storied building, you would really even dream of going back to the builder and say that I need a new sub-basement that would be. There would be a question which most people will love it because of the cost because of risks because of the difficulty of doing that.

But in software similar requirements will very routinely ask for that so the fact that the software can be changed goes severely against the software that a product never can get remain stable and continue to deliver the goods that is required. So this flexibility adds a lot of complexity to the whole process of software development.

(Refer Slide Time: 06:03)




Finally is a problem of characterizing the behavior of discrete systems which adds a lot of complexity. Initial computing systems were mechanical which are very simple after electrical sciences came under our command and elementary electronics was possible, then we came up with lot of computing which is known as analog computing where basically you had analog systems doing the computation for you but really the evolution of computing happen when we move to digital computers that is we do not have continuous value.

So if we if I throw a ball in air and track the trajectory I would not expect the ball suddenly to start

going high at the top of the height or suddenly start going straight we will expect the ball to have a parabolic path but if you are simulating that same process in your digital computer, it is quite possible that you will see this kind of behavior. Why does it happen, why does it go that way because the software being build and executed by a digital system is modeled as a discrete event system.

I will not go into the depth of what discrete different systems are. They are more of study of system dynamics, what it simply means that if we talk in terms of automata theory, it has caught a finite set of states and when it works, it simply changes from one state to the other. Very nice so actually in that way the things should have been simple because an analog computing will have infinite number of states but there are 2 difficulties.

1 analog computing has strict mathematical model if I try to compute the trajectory of the ball through analog means I will use simple equations derived from newtons laws and they will divide everything. But if I have to do it through digital means, I need mathematics, I need tools which are very different from the some ordinary physical world laws and not all of those yet are very clearly understood. The second is of course discrete systems have predefined well-defined states.

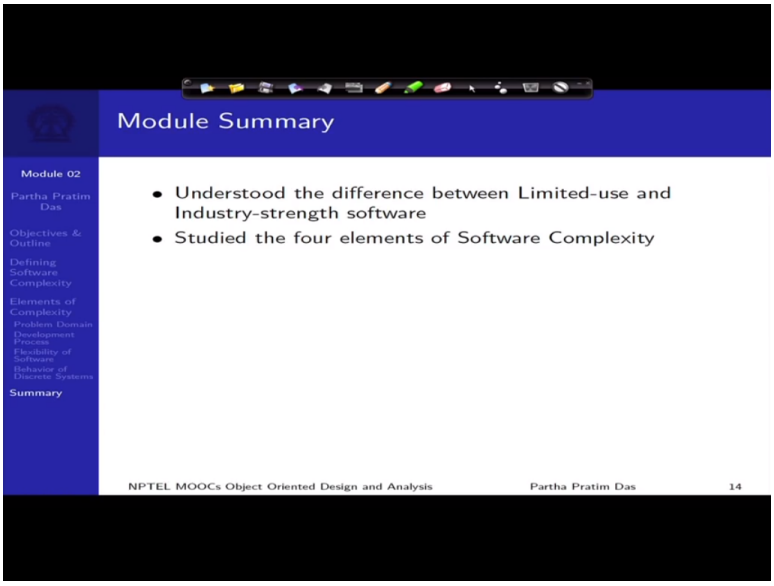
But if you consider a large application, then the application will have 100s of 1000s of variables and possibly in today's time multiple threads of control, multiple things can be happening at the same time given that the state of a system is the entire collection of these variables, the value that the variables has, the current address of each and every process, the current stack of each and every process and so on. So even though in theoretical terms, software has infinite number of states.

Yet many of these states are intractable we just do not understand what these states are, how to enter into those states, what to do once your system gets into that state, how to get out of that. Even at times it is very difficult to even identify that I have got into a state which is not where I want to be in. So take all those states together and say okay this is this is where the software is not working this is where the software is crashed, this is where we have an exception and so and so forth.

But the bottom line is there are several external factors which decide how your system run makes transition between these large number of states and the total behavior of the resulting discrete system that we make as a software is never understood. The tools today do not exist for doing that the skill does not exist with most of the developers to analyze them. So it adds a several dimension of

complexity to the software because of this behavior of discrete systems.

(Refer Slide Time: 10:41)




So these are 4 different elements of software complexity. In this module we have started by trying to understand the difference between limited use personal kind of software and industrial strength software and then we have take a tour of the 4 fundamental elements which make software complex that the complexity that arise from the problem domain from the communication gap, the external complexity, the complexity that arise due to the difficulty of managing big teams which need to be managed.

They are the machines of software development, the flexibility that the software offers is the strength of the software but we have seen why that adds to the complexity of software development and finally the software being a discrete system, the behavior of the discrete system adds a new dimension to the complexity of software. So having understood this, we will stop now. In the next module, will come up and start understanding about what the structure of complex systems typically are and how to manage those complexities.