Software Conceptual Design

Dr. Sridhar Iyer
Dr. Prajish Prasad
Dr. T. G. Lakshmi
Department of Computer Science and Engineering
Indian Institute of Technology, Bombay

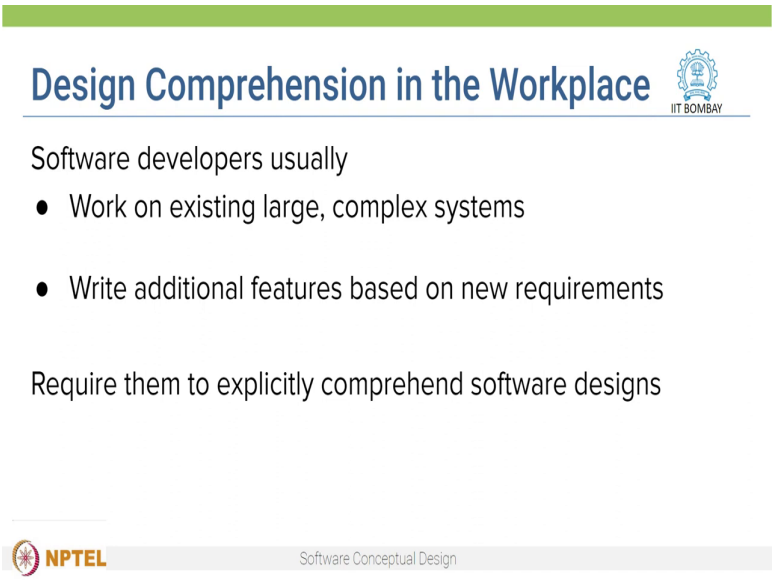
Lecture - 17
Software Design Comprehension

In the previous video, we looked at different UML diagrams used to model a given design. We saw that a given design can have various diagrams which describe multiple views of the system at different levels of granularity.

So, we can understand the software system through these detailed design diagrams, is not it?

Yes, this process of trying to understand a given design is known as software design comprehension. Design comprehension is an important skill required in the workplace.

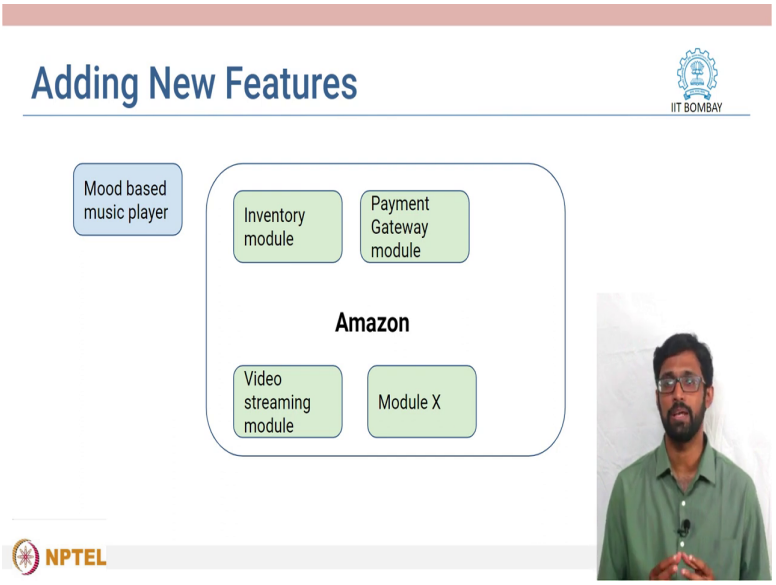
(Refer Slide Time: 00:51)



Software developers usually work on large, existing complex systems, and they write additional features based on new requirements given to them. These tasks require them to explicitly comprehend software designs.

Let us think of a scenario, where design comprehension is needed.

(Refer Slide Time: 01:18)

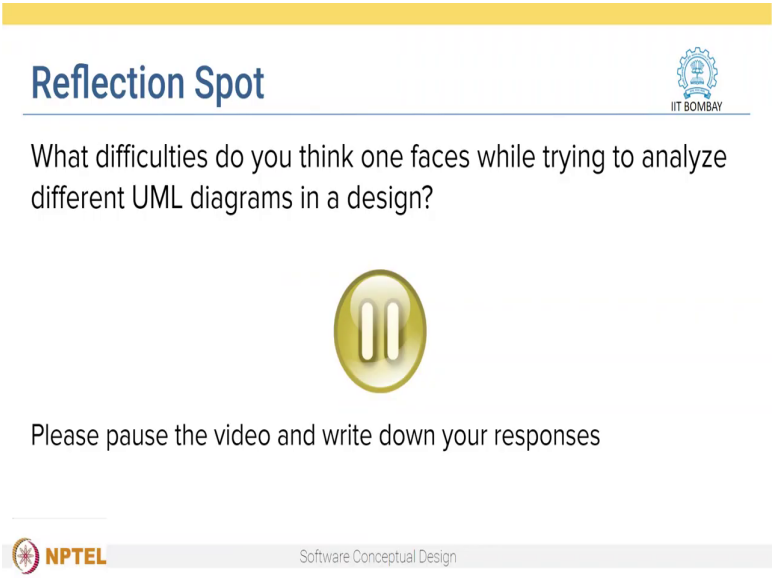


For example, let us say we need to integrate the mood based music player with an existing Amazon music application. How do we go about doing this? So, developers, before they actually start coding, they need to comprehend existing subsystems in the music player and design the new mood based music feature. So that it can be seamlessly integrated to the existing design.

I understood that design comprehension is essential, but let us say I am a student was just learnt about these UML diagrams. It can be difficult to make sense of all these diagrams right?

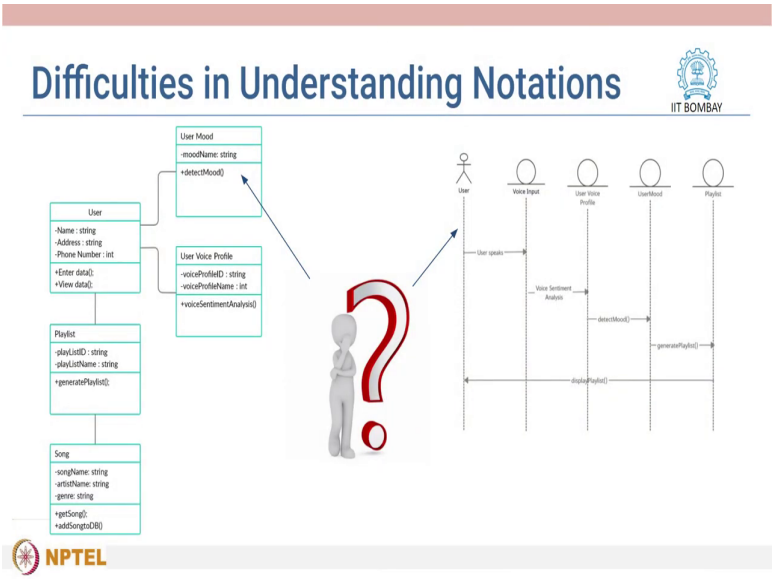
Yes, that is true. Let us reflect on these difficulties. So, students, here is a reflection spot for you.

(Refer Slide Time: 02:24)



What difficulties do you think one faces while trying to analyze different UML diagrams? You can pause this video and write down your responses before proceeding.

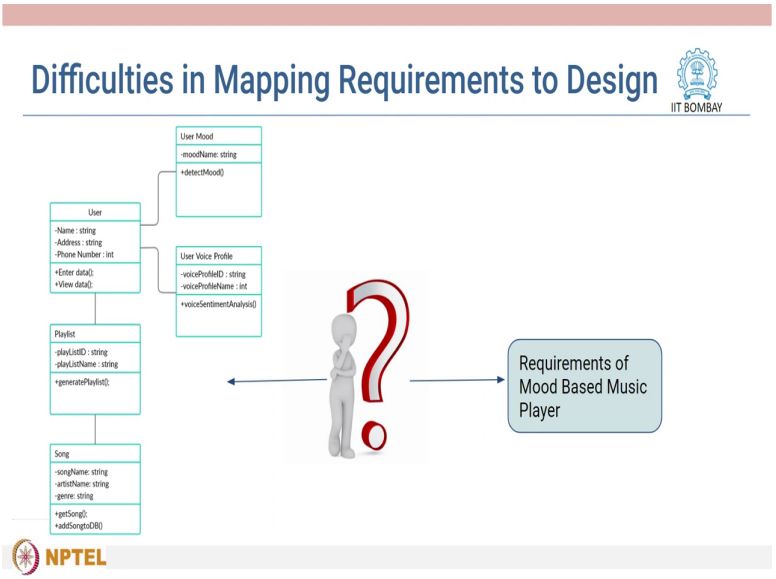
(Refer Slide Time: 02:45)



One might not understand what these diagrams mean. For example, there are various syntax and notations for the different design diagrams.

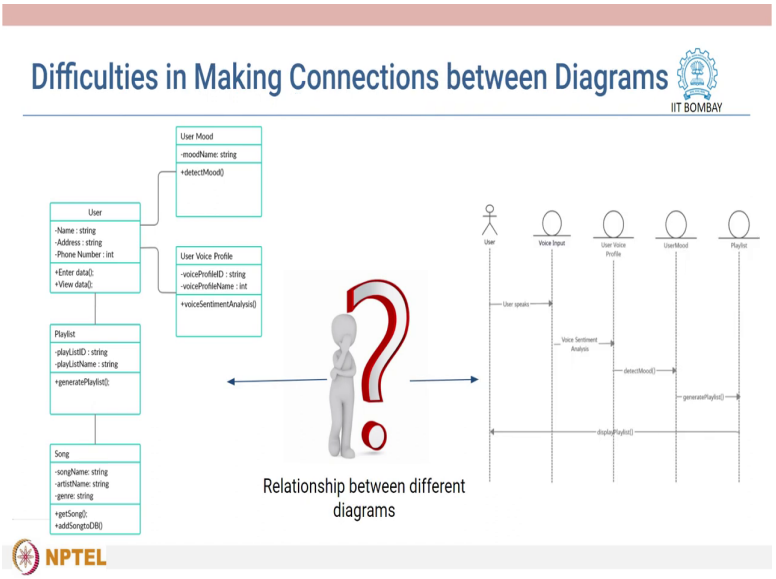
Yes, that is true. But this can also be easily learned and practised.

(Refer Slide Time: 03:07)



Others might find it difficult to understand how certain requirements and parts of the solution design map. For example, how do classes in the class diagram, like song similarity, etcetera relate to the requirement of the mood based music player?

(Refer Slide Time: 03:30)



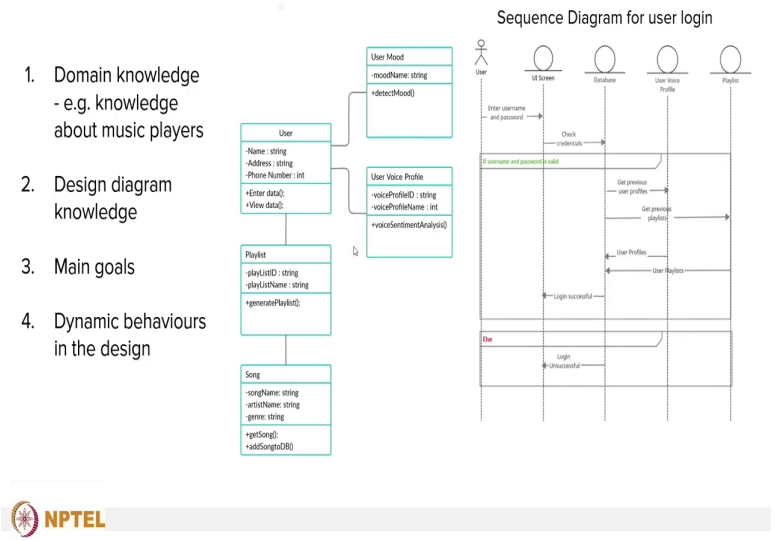
Another difficulty that people could face is making connections between these diagrams, mainly because each of the diagram has huge amount of details.

Yes, that is true. For example, the sequence diagram takes objects from the class diagram and describes its behaviour. However, students find it difficult to make this connection between the class diagram and the sequence diagram.

But it is important to make connections between the different UML diagrams to comprehend the design, but how can we develop design comprehension?

Yes, this is a very good question. To answer this let us see how an experienced software designer tries to comprehend different diagrams in the design. So, how does an experienced software designer comprehend a given design?

(Refer Slide Time: 04:37)



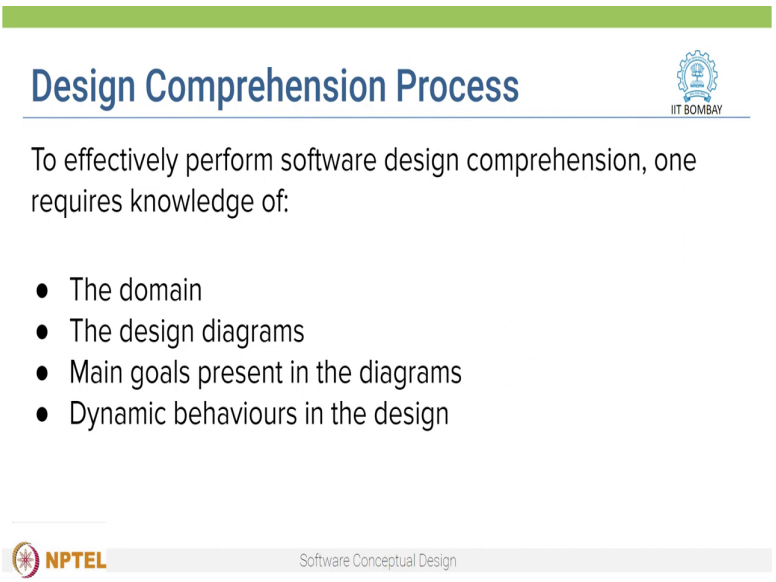
Let us take the example of the design of the mood based music player, while trying to comprehend this design, one needs, first of all, an understanding of the domain. Problem domain knowledge refers to the knowledge about the context for which the design has been developed.

For example, in the case of the mood based music player, students should be familiar with how a music player works. One also needs knowledge about the design diagrams itself. The design diagram knowledge is based on the syntax and semantics of different design diagrams. For example, students should understand various symbols and representations in the class and the sequence diagrams, and what they mean in the context of software design.

One should also understand the main goals of these design diagrams. The main goals refer to the purpose of each diagram and its role in the design. For example, a sequence diagram for a particular functionality can be composed of a sequence of sub goals that realize that functionality. For example, a sequence diagram for login into the mood-based music player will include sub-goals such as user input, functionality is to do when the login is successful and what to do when the login is unsuccessful.

One also requires knowledge about the dynamic behaviours in the design. When expert designers see a set of design diagrams, they think about the dynamic behaviours in the design. This involves knowledge about how data members in the class diagram change based on the execution of the sequence diagram. For example, when a login is successful, a user's profile and playlists are fetched from the database, and corresponding values of the user profile and the playlist should be updated.

(Refer Slide Time: 07:11)



To effectively perform software design comprehension, one requires knowledge of : the domain, the design diagrams, the main goals present in the diagrams and the dynamic behaviours in the design, which is nothing but how the system works in real time.

Ok. So, that is how experienced software designers make sense of each of these diagrams, but I wonder if there is a way to help students comprehend such diagrams.

Yes, the Verisim learning environment helps in just that. Verisim is a learning environment which helps students effectively comprehend various design diagrams. We look at the activities and features of Verisim in more detail in the next video.