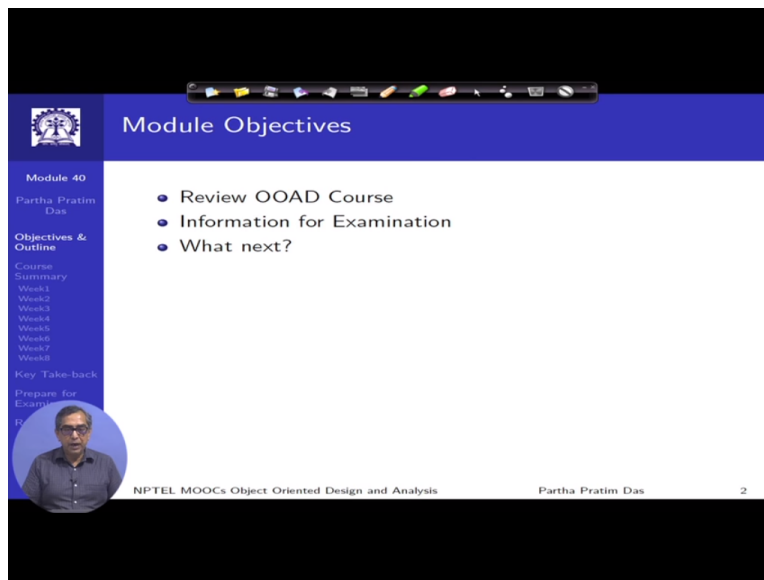


**Object-Oriented Analysis and Design**  
**Prof. Partha Pratim Das**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology-Kharagpur**

**Lecture – 52**  
**Closing Comments**

Welcome to module 40 of object oriented analysis and design. As you know this is going to be the last module of the course. So I would not discuss about any new topic, new aspect of object-oriented analysis and design.

(Refer Slide Time: 00:49)



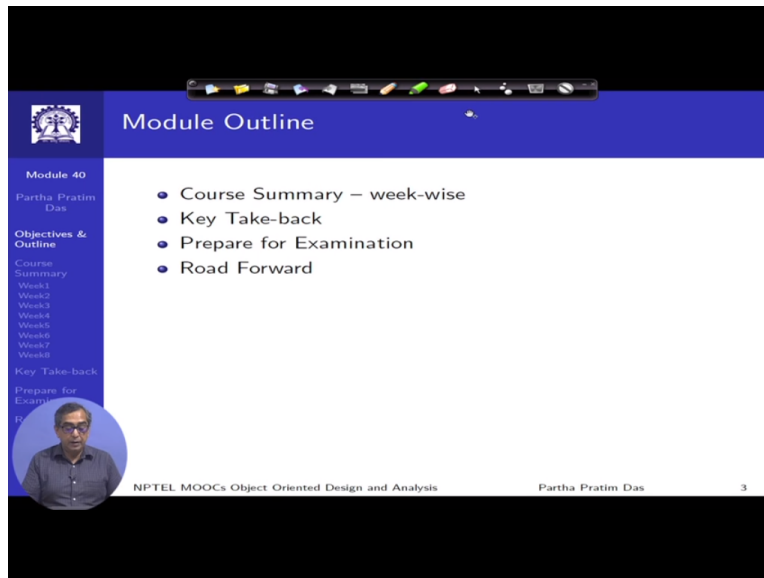
The screenshot shows a presentation slide titled "Module Objectives" for "Module 40". The slide is part of the "NPTEL MOOCs Object Oriented Design and Analysis" course by "Partha Pratim Das". The slide lists three objectives:

- Review OOAD Course
- Information for Examination
- What next?

The slide also includes a sidebar with navigation links: "Module 40", "Partha Pratim Das", "Objectives & Outline", "Course Summary", "Week1", "Week2", "Week3", "Week4", "Week5", "Week6", "Week7", "Week8", "Key Take-back", "Prepare for Examination", and "Review". A small circular portrait of the professor is visible in the bottom left corner of the slide.

Rather I will spend a little while putting some closing comments. So the objective is to review briefly what we have done in this course and mentioned to you why you should be preparing for your examination and beyond this course what you should be looking at.

(Refer Slide Time: 01:03)



**Module Outline**

- Course Summary – week-wise
- Key Take-back
- Prepare for Examination
- Road Forward

Module 40  
Partha Pratim Das

Objectives & Outline  
Course Summary  
Week1  
Week2  
Week3  
Week4  
Week5  
Week6  
Week7  
Week8  
Key Take-back  
Prepare for Examination  
Road Forward

NPTEL MOOCs Object Oriented Design and Analysis Partha Pratim Das 3

So this forms the basic outline of the module.

(Refer Slide Time: 01:04)



**Week 1: Software Complexity**

- **Is Software Engineering? Why Software Projects Fail?**
- **Four Elements of Complexity:** Complexity of the Problem Domain, Difficulty of Managing the Development Process, Flexibility Possible through Software, and Problems of Characterizing the Behavior of Discrete Systems
- **Structure of Complex Systems**
- **Attributes of a Complex System:** Hierarchic Structure, Relative Primitives, Separation of Concerns, Common Patterns, and Stable Intermediate Forms
- **Organized Complexity:** Decomposition Hierarchy, Abstraction Hierarchy, Class and Object Structure, and Canonical Form
- **Disorganized Complexity:** Limited Human Capacity
- **Bringing Order to Chaos:** Role of Decomposition, Abstraction, and Hierarchy
- **Designs and Models**

Module 40  
Partha Pratim Das

Objectives & Outline  
Course Summary  
Week1  
Week2  
Week3  
Week4  
Week5  
Week6  
Week7  
Week8  
Key Take-back  
Prepare for Examination  
Road Forward

NPTEL MOOCs Object Oriented Design and Analysis Partha Pratim Das 4

So if we look back and I would take a quick review based on how the modules were organized in terms of different weeks and moving from one week to the next. In most cases we tried to achieve something foundational about the course, about the discussion. So we started off with what is the software complexity all about. So we started with a very basic question of is software at all an engineering and reasoned to come to the conclusion that we still develop software.

We are far away from actually constructing software, the way several engineers construct buildings. But in the process so we identified some of the fundamental reasons as to why the software projects fail and this lead us to identify four elements of complexity that really plague the software. Complexity of

the problem domain, difficulty of managing the development process, flexibility possible through software, this appears to be an advantage and goes against the software and the problem of characterizing behavior of discrete systems and this is the reality that will have to work with.

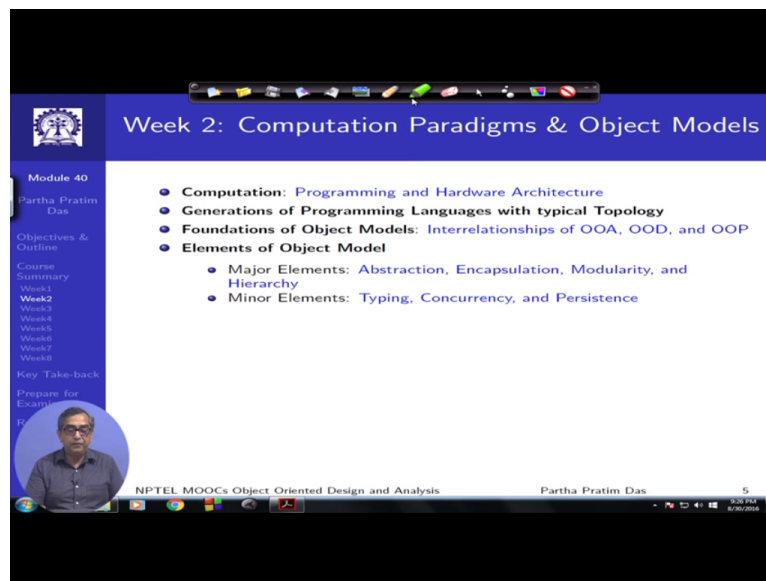
So we then try to look into how the complex systems are structured, what is their generic structure and we observed looking at natural, man-made, social different kinds of system that there is a whole lot of commonality between the structure of complex systems and that can be leveraged to really come up with designs or come up with the design approaches which can prove to be really good in terms of software development.

So we identified the attributes of complex systems and 5 attributes were identified hierarchic structure, relative primitive, separation of concerns, common patterns and stable intermediate form and this lead us to discuss about organized and disorganized complexity and as we have seen that these organized complexity became the source of main study in the object-oriented analysis and design that the organized complexity manifest itself in terms of decomposition and abstraction hierarchy, class and object structure and can lead to what is known as a canonical form of a system representation.

We also recognize that there are other forms of complexity which is disorganized which is primarily due to the limitation of the human brain and we can hardly do much about this other than doing the organized, managing the organized complexity better. Now finally to bring an order to the chaotic scenario of complex systems, we argued that we need to look at decomposition, abstraction and hierarchy. These are this will be the key tools by which we can design and model the system.

So this was the basic analyzing the foundational reality of software and the reality of real-world systems, the complex systems was a main take back from the study of software complexity and object-oriented solutions for them in weak object.

(Refer Slide Time: 04:47)

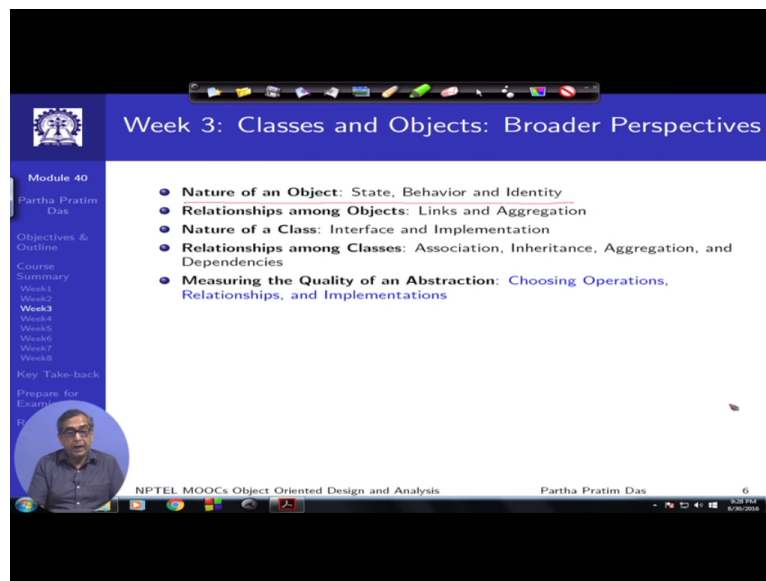


In week2, we took a look little bit differently in terms of what is among if the software is complex if the systems are complex then we wanted to also look at what is in terms of the computational paradigms and object models and we look we can deep look into different programming and hardware architecture, the generations of languages how languages have evolved, how programming paradigms have evolved, what has been the typical topology and so on.

And along with that we introduced the foundations of object model which we have did it, we have done it in three stages saying one part is object oriented analysis, then you have object oriented design and finally you culminate all that into object oriented programming which were your actual software system gets realized and in terms of been that we we could identify that there are 4 elements in a object modeled the abstraction, the encapsulation, the modularity and the hierarchy and also there are 3minor elements.

So these this point onward these elements have become the guiding factor of deciding on how we do the analysis, the object oriented analysis how you actually do it, going to doing a design which is efficient which is manageable for a complex system. So that was the primary primary aspect to discuss in week 2.

(Refer Slide Time: 06:25)



In week 3 we moved closer to the commonly known notions of object oriented terminology, we talked about the nature of objects and identified that an object is something that exists as a state, as behavior and as identity and we deliberated at length in terms of what and how is an object and we extended that for relationships among objects we showed that the objects just existing by themselves does not provide anything interesting but they become interesting when they have links when they get aggregated between themselves.

While objects are interesting in terms of actual programming we need a layout we need a template through which we need to express these all these object behavior. So next was the discussion of the nature of classes, particularly in terms of the interface and implementation. We saw that the interface is how the objects instantiated from that class will interact with others and implementation is how actually the object of that class come into being how actually the behavior, the state and behavior of the object will be managed.

And both of these are pretty important to be able to finally realize the objects of an object-oriented programming system. Relationships amongst classes give us a lot of direction in terms of how should the modeling be done what are the different modeling possibilities that exist in terms of the real world behavior in terms of the real world connectivities and in this we also took a different measure of looking into that since all of these are objects, classes all are different forms of abstraction of the real world.

Because certainly the real world was not created to with the stamp of this is this object or this is the class, this is the relationship and so on. So when we identify such objects and classes what should be

our measure of quality of abstraction when we say that the obstruction is good when we say the abstraction is not so good and has scope for improvement so we talked about measures of quality of abstraction, in terms of choosing relations operations, choosing relationships and choosing implementation?

(Refer Slide Time: 09:10)

Week 4: Classes and Objects: How to Identify?

Module 40  
Partha Pratim Das  
Objectives & Outline  
Course Summary  
Week1  
Week2  
Week3  
Week4  
Week5  
Week6  
Week7  
Week8  
Key Take-back  
Prepare for Exam  
R

- What is Classification
- Extraction of Classes
  - Classification: Structural Clustering, Conceptual Clustering, and Prototyping
  - Identification: Key Abstractions
- Classes, Objects, and Relationships in LMS: Linguistic Approach and Analysis

NPTEL MOOCs Object Oriented Design and Analysis Partha Pratim Das 7

And we then made this a tool in terms of carrying out a detailed exercise on our part to be able to show you that how we can put all these discussion of object-oriented analysis into practice. So the basic question that we started to address in week 4 is good that we realize that object oriented approach will give us a much better alignment with the inherent structure of complex systems and they should be organized in terms of objects and classes and their behaviors, relationships and so on.

But that is still leaves one basic fundamental question unanswered as to given a real world situation how do I identify the classes, how do I identify the objects. So we talked about the basic classification process to help the extraction of classes and we learnt about 3 basic very fundamental foundational methodology for classification in terms of structural clustering, conceptual clustering, and prototype based classification or prototyping and we also learned about how we do key abstractions for identification.

And these become effective tools at our hand to look at a real world situation at length and actually try to extract the classes, objects relationships, hierarchies, refinements, dependencies and so on. So we spent a major part of week 4 in trying to apply all these in the context of a running system example which we had introduced in I think in week 1 or week 2 early in the course a leave management system

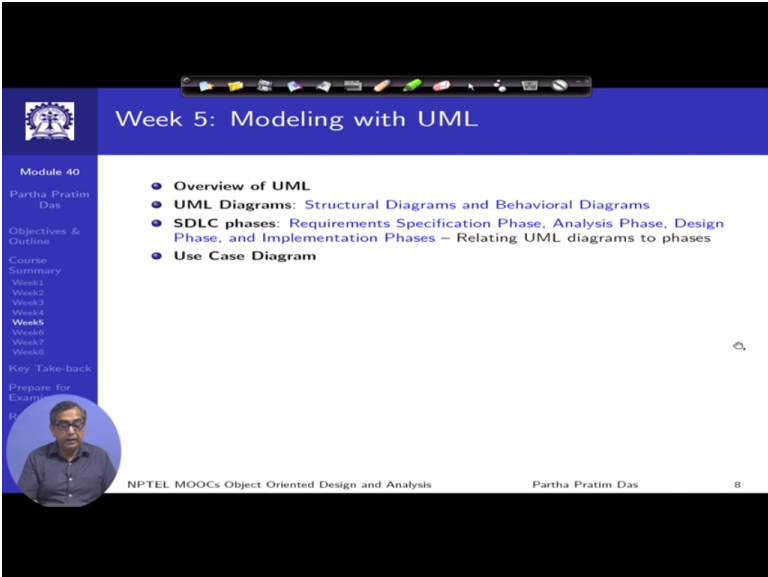
and we demonstrated with one special part of the approach which is a linguistic approach.

There could be other approaches as well primarily trying to analyze the different parts of speech of a given English description of a system and reference to certain video based interaction between the customer and the client we try to analyze and identify nouns, verbs, different parts of speech to lead us to the different classes their relationships their hierarchies and so on and every time we worked on on such a model identifying classes, objects, relationships we tried to get back to the quality parameters that we have defined for the abstraction.

And measure as to are we really improving from the earlier model and in that process we shoed through a series of refinements that possibly went over 3 or 4 modules that starting from a 2 page English description we are able to come out with a significant number of classes, their hierarchies are major attributes, their relationships and put them and and those more or less everything is has certainly the dimensions of who is doing it, what is the expertise of the person analyzing.

But still we could identify several techniques by which more or less uniformly all these information can be extracted.

(Refer Slide Time: 12:37)



The screenshot shows a presentation slide for 'Week 5: Modeling with UML'. The slide has a blue header with the title. Below the header, there is a list of topics: 'Overview of UML', 'UML Diagrams: Structural Diagrams and Behavioral Diagrams', 'SDLC phases: Requirements Specification Phase, Analysis Phase, Design Phase, and Implementation Phases – Relating UML diagrams to phases', and 'Use Case Diagram'. On the left side, there is a sidebar with a navigation menu. The menu includes 'Module 40', 'Partha Pratim Das', 'Objectives & Outline', 'Course Summary', 'Week 1', 'Week 2', 'Week 3', 'Week 4', 'Week 5', 'Week 6', 'Week 7', 'Week 8', 'Key Take-back', 'Prepare for Exam', and 'Review'. At the bottom of the slide, there is a footer with the text 'NPTEL MOOCs Object Oriented Design and Analysis', 'Partha Pratim Das', and the page number '8'. There is also a small circular portrait of Partha Pratim Das in the bottom left corner of the slide.

And that was the basic objective of the week 4 which culminated the first half of this course with the introduction of the object-oriented paradigm as a whole and culminating with certain practical approaches to identifying objects, classes and relationships within the system. The next half of the course covering week 5 to the week 8 has been dedicated to the other aspect of the course which is

once the information once the object-oriented components, classes, object, relationships hierarchies have been extracted the different process have been identified.

Different use cases have become known, how do we represent them in terms of a language which is not biased by a particular programming style, a particular platform or a particular individual a particular process and so on and in that context we introduced what is known as a unified modeling language on uml and we discuss we provided started with overview showed that uml did not happen in a day, it went to a evolution of several years when different concepts.

Finally got unified in terms of a uml collection of uml diagrams which are classified as structural which talks about basically the levels in which different part of the system exists and the behavioral diagrams which say how the system will behave over period of time now to combine these 2, the first part where we identified the basic object-oriented features of a system, the classes behaviors object behaviors, relationships and so on and their expression through such a standardized language to be able to bridge this gap.

We also took a detour into the basic sdlc process which is a software development lifecycle process. So we wanted to say that well it is good that we have a language which is a platform agnostic, which is process agnostic, which is people agnostic and so on and we know how to extract different object system information from the real world through the methods that we did in the first part but when do you do what? I the uml has several diagrams which diagram should I use at which stage of software development? what am I?

What information should I be interested or focused on extracting at different stages? So we took a tour in terms of the sdlc phases primarily the part of that requirements to analysis to design which is significantly these are the 3 where OOAD is significantly involved but it spills on to the high level low level design aspects the object-oriented programming aspects in the implementation phase, we also talked about this, we have not considered the other phases like the actual deployment of the system or the maintenance and the overall other lifecycle aspects or the testing phase and so on.

They also involve object oriented analysis and design but certainly in a less significant way than these phases and with this equipped with all these, we have dedicated a significant amount of I mean more that I think 40 percent of the time of this course in actually explaining understanding the uml diagrams



and we have done that in the kind of perceived order of their importance and how widely you are expected to use them.

So we started with the use case diagram which is certainly the most foundational which you use in the requirements phase and you will need to use that every time you come across a system.

(Refer Slide Time: 16:54)

The slide is titled "Week 6: Modeling with UML" and is part of "Module 40" by Partha Pratim Das. It lists the topics for the week: "Class Diagram" and "Sequence Diagram". The left sidebar contains a navigation menu with options like "Objectives & Outline", "Course Summary", and a list of weeks from Week 1 to Week 8, with "Week 6" currently selected. A small circular video inset shows the presenter, Partha Pratim Das. The footer includes the text "NPTEL MOOCs Object Oriented Design and Analysis" and the slide number "9".

And then we moved on to a structural diagram the class diagram which is again possibly the most widely used uml diagram which also leads to the direct consequence in terms of object-oriented programming, then we moved on to sequence diagram which talks about the processes and algorithms particularly in the temporal domain.

(Refer Slide Time: 17:18)

The slide is titled "Week 7: Modeling with UML" and is part of "Module 40" by Partha Pratim Das. It lists the topics for the week: "Communication Diagram", "Activity Diagram", and "Interaction Overview Diagram". The left sidebar contains a navigation menu with options like "Objectives & Outline", "Course Summary", and a list of weeks from Week 1 to Week 8, with "Week 7" currently selected. A small circular video inset shows the presenter, Partha Pratim Das. The footer includes the text "NPTEL MOOCs Object Oriented Design and Analysis" and the slide number "10".

And continued to discuss about different behavioral diagrams in terms of the communication diagram which is the communication diagram which talked about the special information of messages and also we kept regular reference to what is happening in terms of the original computing model which in our assumption has been the client-server model. We talked about the activity diagram which is the sequencing of operations as identified from the different use cases the interaction view diagram which is another smaller diagram.

(Refer Slide Time: 17:56)

The screenshot shows a presentation slide with a blue header and a white main content area. The header contains the text 'Week 8: Modeling with UML' and a small logo on the left. The main content area is divided into two columns. The left column is a blue sidebar with white text listing the course structure: 'Module 40', 'Partha Pratim Das', 'Objectives & Outline', 'Course Summary', 'Week1', 'Week2', 'Week3', 'Week4', 'Week5', 'Week6', 'Week7', 'Week8', 'Key Take-back', 'Prepare for Exam', and 'Review'. The right column has a white background with a list of UML diagrams: 'State Machine Diagram', 'Timing Diagram', 'Component Diagram', 'Deployment Diagram', 'Composite Structure Diagram', and 'Package Diagram Diagram'. A small circular inset image of a man is visible in the bottom left corner of the slide. At the bottom of the slide, there is a black bar with white text: 'NPTEL MOOCs Object Oriented Design and Analysis', 'Partha Pratim Das', and '11'.

And then finally we in the current week we ended up with the state machine diagram which is again a very important diagram giving us lot of state related information in the system and its more powerful than the finite state machine, it basically gives a state chat information and subsequently just in the last module we took a quick look into some of the other uml diagrams which are also very important but in the context of OOAD they may have less significance because they relate more closely to the implementation phases than the analysis and design phases.

(Refer Slide Time: 18:40)

**What have we learnt?**

- Software is Complex because Systems are Complex
- OOAD helps understanding and working with Complex Systems because
  - OOAD is natural and scalable
  - OOAD is process, people and platform agnostic
- UML is a powerful modeling instrument that
  - provides a language of expression that is process, people and platform agnostic
  - aligns perfectly with object-oriented paradigm
  - works at multiple levels of details with multiple facets
- **Object-Oriented** is not in the tool, language or platform – it is how one looks at a system – it is in the **Analysis** and the **Design**!

**Object-Oriented is in the mind of the Developer**

Module 40  
Partha Pratim Das  
Objectives & Outline  
Course Summary  
Week 1  
Week 2  
Week 3  
Week 4  
Week 5  
Week 6  
Week 7  
Week 8  
Key Take-back  
Prepare for Exam  
NPTEL MOOCs Object Oriented Design and Analysis  
Partha Pratim Das  
12

So having looked at that later if we just look back turn back and in one slide I want to summarize as to what have we learned? I think this will be the core learning that we are looking at. Certainly the first message we have learnt is software is complex because if you leave a side everything and anything else it is complex because systems are complex that is an inherent panes of that. So there is no compromise on this to the first lesson we have learned.

Second is object-oriented analysis and design helps understanding and working with complex systems. Why? Primarily there are several reasons but primarily for 2 reasons, one is object-oriented analysis representation design is natural, it is not a synthetic approach, it is something that you see around happening in your real world and you adjust imbibing that in your process and that is the reason that there is such a great alignment between the structures of complex systems as we saw.

And the different primitives that the object-oriented analysis and design have started to offer. It is scalable that it works for a very small system a payroll system of 10 employees to a mass learning systems if I may talk of everywhere object-oriented analysis of design work in every scale so that gives a major reason as to why it should be used it is used in almost every kind of complex system. Second fundamental reason is OOAD is a process, people and platform agnostic.

The whole approach does not depend on individual, does not depend on what process you want to follow or which platform you are trying to use by platform I mean software, hardware, physical systems everything. So OOAD it is a very very generic approach. The third take back I would say is a uml we have learnt uml as a very powerful modeling instrument and it is powerful for several reasons.

One is it provides a language of expression that again is process, people and platform agnostic.

It does not say that you have to use C++ or java, it does not say you have to work on unix, it does not say that you will have to use agile or waterfall or rational rise process, none of that. You could be you could do anything in terms of all these different choices of design and exercise but you have a language which is standardized which in many cases is executable and which certainly across both the community will understand. Second reason that uml stands your is it aligns very perfectly its designed very perfectly in alignment with object-oriented paradigm.

So whatever we want to say or whatever you identify in terms of object-oriented and design that can be expressed very aptly, very suitably in terms of the primitives. So that is a great reason as to why uml is a powerful modeling instrument. Finally it is powerful because it can work in multiple levels of details and multiple facets. What does that mean is there is uml when you are at the requirements phase where you almost know nothing about the system.

So you are at a very abstract, very vague level of details. Then you slowly capture certain diagrams, you capture use case some basic class diagram and then you start refining them. So as you keep on refining them, you are increasing on the level of details further and further and further and you continue to do that and uml will continue to support you and it is multifaceted because it is you talk about time, you have sequence diagram, timing diagram, you talk about space, you have collaboration diagram.

Component diagram, you talk about object hierarchy, you have object diagram, class diagram. So you may want to look at the system from multiple different angles and uml today tries to give you a ability to look at a system, represent a system, expresses system from almost any angle that you feel comfortably. So the third take back in my view is your understanding and you know ability to use uml as a modeling instrument. Finally I would like to say that object orientation is not a tool, language or platform. It depends actually on how one looks at the system.

I am all often faced with this question is, is it necessary to use say C++ to be able to design and implement object-oriented system? Can I use c and do that? I often would say that yes, the question is object-orientation is not significantly in that, will certainly the tool helps certainly the language helps but you can still do a very good object-oriented design in a language like c which has zero object-oriented feature provided you can look at the system in a object-oriented manner, you can look at the

system with the proper object-oriented analysis and design approaches.

So to summarize in terms of the take back this is my one line recommendation to you all is object-orientation please keeping mind is in the mind of the developer. uml will help analysis approaches will help the language will help, certain tools, rational lows visual paradigm, they will help but whether how well you have actually analyzed and design a system in object-oriented terms depend on how as a developer you think object-oriented, how any concept that you come across, how quickly you can identify, see through and identify what is a key abstraction.

How quickly you can see that abstraction, how quickly you can see through that what is an appropriate state, what is the appropriate set of behavior for an object, what is should be a right hierarchy of specialization for this. The more of this you can reason out in your mind, you become a better programmer, you become a better developer for the object-oriented paradigm and  
(Refer Slide Time: 25:54)

The screenshot shows a presentation slide titled "Prepare for Examination" for "Module 40" by Partha Pratim Das. The slide is divided into a left sidebar and a main content area. The sidebar lists "Objectives & Outline" with a list of weeks (Week1 to Week8) and a "Key Take-back" section. The main content area lists tasks: "Watch the Videos", "Revise the Assignments and Solutions", "Practice Used / Worked Out Examples", and "Practice with Additional Examples". Under "Practice with Additional Examples", it lists: "(Indian) Postal Management System (PMS)", "(Newspaper) Story Management System (SMS)", and "(Rental) Car Management Systems (CMS)". Under "Study Books", it lists: "Object-Oriented Analysis and Design – With Applications by Grady Booch, Robert A. Maksimchuk, Michael W. Engle, Bobbi J. Young, Jim Conallen, Kelli A. Houston, 3rd Ed., 2007" and "Learning UML 2.0 – A Pragmatic Introduction to UML by Russ Miles, Kim Hamilton, 2006. Publisher: O'Reilly Media". The slide is part of an NPTEL MOOCs presentation on Object Oriented Design and Analysis.

That I would emphasize is a key take back which you should focus on as you barge into the last couple of weeks of reading and practicing on this course to specifically prepare for your examination I will have a very classical advise that you certainly watch the videos, revise the assignments and solution. There is nothing new to that but certainly I would strongly advise that you practice that used and work out examples, this example I have used very extensively all through the presentations.

And this example has often been referred to in terms of the assignment. So I would strongly advise that you practice on these examples more and more, practice, try to do all kinds of diagram for them, try to

do all kinds of analysis for them, try to look into the systems in multiple different ways that you would like to do in a multi-faceted manner, in multiple levels. Besides these are some of the systems for which we will have prepared the small system descriptions, postal management system; story management system and a car management system.

So these are also like one and a half 2 page description. So we will soon upload this pdf files and I would suggest that you can use them as additional examples. if you have practiced with the leave management system watching the video, the it would be good to pick up one of the systems and try to do the similar analysis and try to build similar models, uml models based on these sample example systems also if you need to refer to books.

Then these are the 2 books certainly this is the most important book the Grady Booches book on OOAD which we have been following from the beginning particularly for the first half of the course where we discuss about different object-oriented concepts, fundamentally much of the material is actually from this book. So it will help if you refer to this book but in addition I would suggest that if you have questions on uml, you could also refer to the learning uml 2.0 I found this quite a nice book to follow.

But just to say along with that there are several sites which have very good information including the uml 2.5 diagram site which for which the URL you will find in several of our presentation slides. So you could also pick up for the uml you could also pick up and work from that.

(Refer Slide Time: 28:21)

The screenshot shows a presentation slide titled "Road Forward" with a blue header. On the left, there is a vertical navigation menu with the following items: "Module 40", "Partha Pratim Das", "Objectives & Outline", "Course Summary", "Week 1", "Week 2", "Week 3", "Week 4", "Week 5", "Week 6", "Week 7", "Week 8", "Key Take-back", "Prepare for Exam", and "Road Forward". The main content area on the right lists three bullet points: "Practice modeling with UML", "Learn **Object Oriented Programming** in C++ or Java or Python", and "Analyze, Design and Implement of moderate sized systems". A fourth bullet point, "Study **Software Engineering** to understand and use processes better", is partially visible. At the bottom left, there is a circular portrait of Partha Pratim Das. The footer contains the text "NPTEL MOOCs Object Oriented Design and Analysis", "Partha Pratim Das", and the page number "14".

Finally going forward that is beyond this course if you would like to study further understand further then certainly kind of things that you can do is can practice modeling with uml. Uml is very very vast as we have seen; it has taken us I mean really about 15, 16 more modules to even discuss introduce you to all these diagrams. So it will take you much longer to practice and get familiar with the nuances of uml and how powerfully you can use that.

But certainly if you do that it will turn out to be something very good in terms of your you as a as a developer, I would say as a developer of software systems besides that it is good when you are looking at analysis and design it is also good to look at the possible ways to implement. So I would strongly advise that you take also take courses in object-oriented programming these are my 3 recommendations in the order of preference I would suggest that you do a course of OOP in C++ or in java or even parallely on python and practice them.

Certainly there is in terms of analysis and design is nothing better than actually being hands-on. so do analyze, design and implement moderate size systems so that you can know through your experiences to what works and what does not work and at a second level, I would also suggest that you can consider studying software engineering more formally which will have a significant part of object oriented analysis and design but you can study software engineering more formally to understand and use processes better in terms of exercising the object-oriented analysis and design practices.

(Refer Slide Time: 30:21)

**Road Forward**

- Practice modeling with UML
- Learn **Object Oriented Programming** in C++ or Java or Python
- Analyze, Design and Implement of moderate sized systems
- Study **Software Engineering** to understand and use processes better

Module 40  
Partha Pratim Das  
Objectives & Outline  
Course Summary  
Week 1  
Week 2  
Week 3  
Week 4  
Week 5  
Week 6  
Week 7  
Week 8  
Key Take-back  
Prepare for Examination  
Road Forward

NPTEL MOOCs Object Oriented Design and Analysis Partha Pratim Das 14

So to summarize I would simply say that the course on object-oriented analysis and design is hereby concluded. I wish you all the very best for your forthcoming examination if you are taking it and

certainly for your future endeavor with software systems analysis, design and the application of the uml diagram as well as the different approaches of OOAD that we have discussed here.