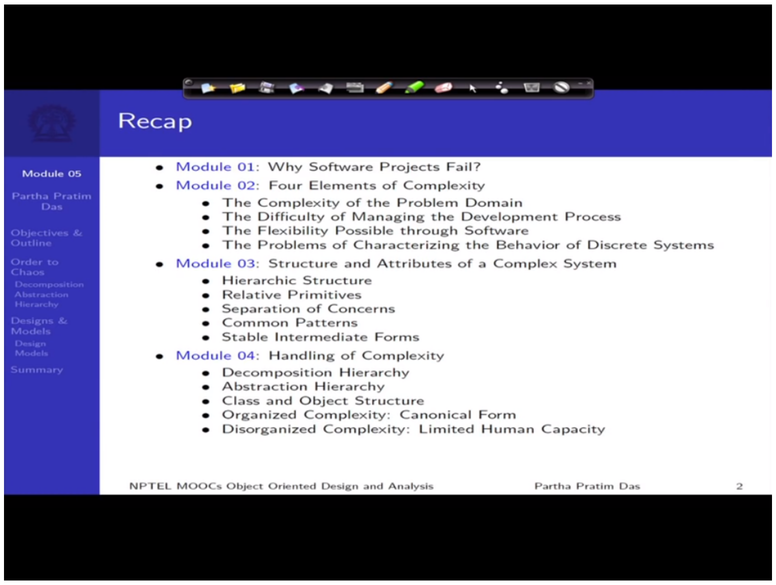
Object-Oriented Analysis and Design

Prof. Partha Pratim Das
Department of Computer Science and Engineering
Indian Institute of Technology-Kharagpur

Lecture – 07
Bringing Order to Chaos

Welcome to module 5 of object-oriented analysis and design. Here we will primarily talk about how what should be the broad approach designing object-oriented systems before I get into the specific objectives.

(Refer Slide Time: 00:41)



Recap

Module 05
Partha Pratim Das
Objectives & Outline
Order to Chaos
Decomposition
Abstraction
Hierarchy
Designs & Models
Design
Models
Summary

- Module 01: Why Software Projects Fail?
- Module 02: Four Elements of Complexity
  - The Complexity of the Problem Domain
  - The Difficulty of Managing the Development Process
  - The Flexibility Possible through Software
  - The Problems of Characterizing the Behavior of Discrete Systems
- Module 03: Structure and Attributes of a Complex System
  - Hierarchic Structure
  - Relative Primitives
  - Separation of Concerns
  - Common Patterns
  - Stable Intermediate Forms
- Module 04: Handling of Complexity
  - Decomposition Hierarchy
  - Abstraction Hierarchy
  - Class and Object Structure
  - Organized Complexity: Canonical Form
  - Disorganized Complexity: Limited Human Capacity

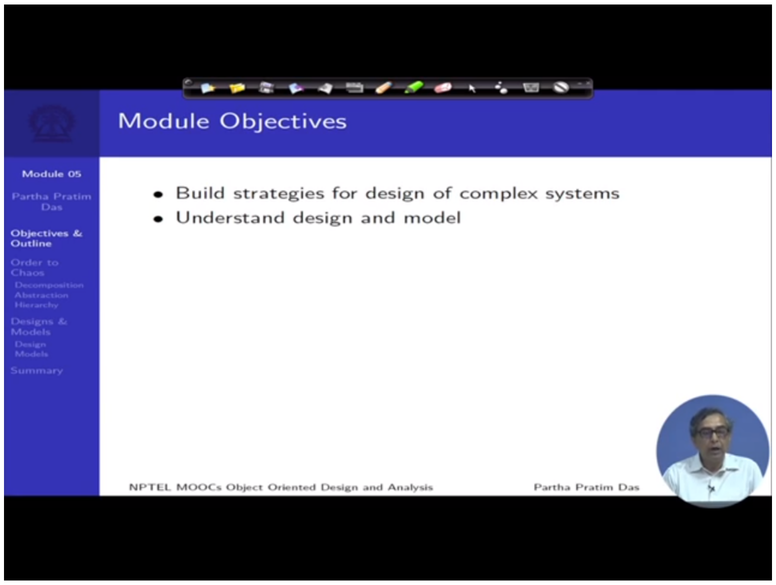
NPTEL MOOCs Object Oriented Design and Analysis — Partha Pratim Das — 2

Let us quickly recap the major points that we have discussed so far in the course. Initially we talked about why software project failed and identified that software development, software projects are inherently complex and there are 4 primary elements of complexity coming from the problem domain, the difficulty of development process managing development process, the flexibility and problems due to the discrete systems structure of the software systems.

Then we try to study different complex systems, natural, manmade, social and so on to understand if there are commonality between difference complex systems and we identified 5 different attributes for complex systems starting from the hierarchic structure, the primitive relative primitives or the building basic building blocks, separation of concerns, common patterns and stable intermediate form. Equipped with this, we did discuss in the last module about how we can handle complexity.

And we observe that the complexity of the design, development process of a complex software system can be divided in terms of organized approach which is primarily driven by a canonical form comprising the object and class hierarchy, the 2 orthogonal dimensions of representation in a complex system and the 5 basic attributes of the complex systems. At the same time, there is a quite a bit of disorganized complexity arising out of the limited human capacity to handle a limited number of information chance at a given point of time.

(Refer Slide Time: 02:47)



Module Objectives

- Build strategies for design of complex systems
- Understand design and model

Module 05

Partha Pratim Das

Objectives & Outline

Order to Chaos

Decomposition

Abstraction

Hierarchy

Designs & Models

Design

Models

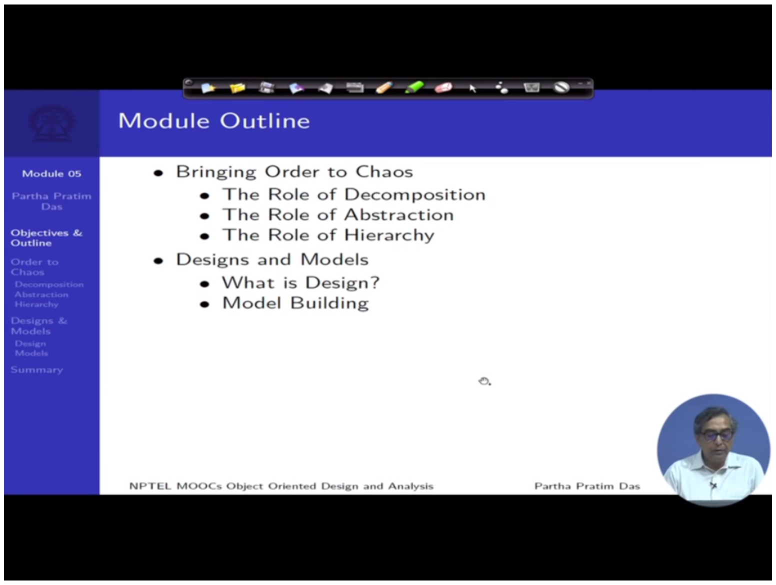
Summary

NPTEL MOOCs Object Oriented Design and Analysis

Partha Pratim Das

So equipped with all these in the current module, we would like to discuss about some basic strategies for design of complex systems. We will define what is our understanding of design and that of a model.

(Refer Slide Time: 03:01)


Module Outline

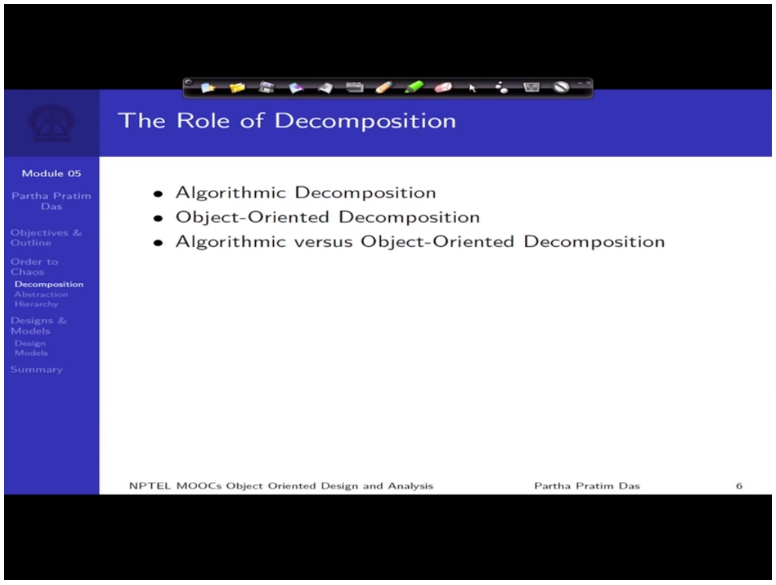
- Bringing Order to Chaos
  - The Role of Decomposition
  - The Role of Abstraction
  - The Role of Hierarchy
- Designs and Models
  - What is Design?
  - Model Building

Module 05
Partha Pratim Das
Objectives & Outline
Order to Chaos
Decomposition
Abstraction
Hierarchy
Designs & Models
Design
Models
Summary

NPTEL MOOCs Object Oriented Design and Analysis Partha Pratim Das

The outline will be primarily about the part to approach the whole design process in terms of the roles of decomposition, abstraction and hierarchy.

(Refer Slide Time: 03:16)


The Role of Decomposition

- Algorithmic Decomposition
- Object-Oriented Decomposition
- Algorithmic versus Object-Oriented Decomposition

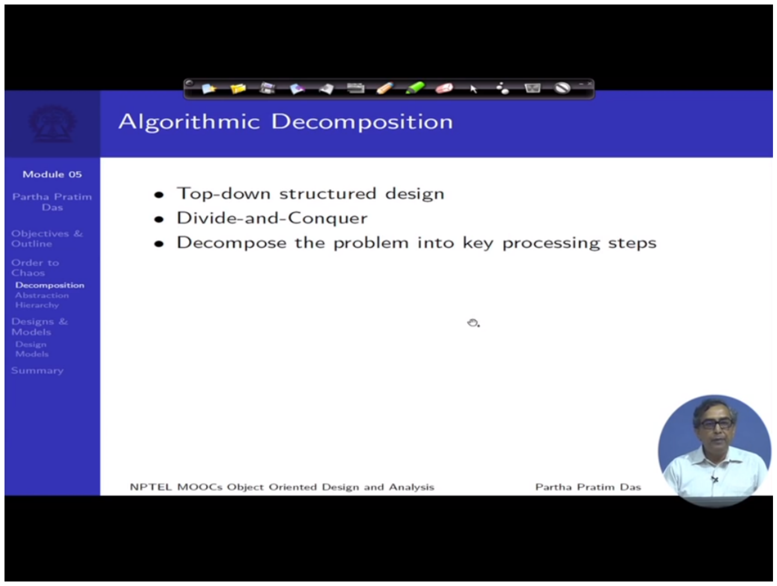
Module 05
Partha Pratim Das
Objectives & Outline
Order to Chaos
Decomposition
Abstraction
Hierarchy
Designs & Models
Design
Models
Summary

NPTEL MOOCs Object Oriented Design and Analysis Partha Pratim Das 6

So we start with analyzing the role of decomposition. As to we should have seen that there is decomposition structure, there is possible decomposition of the system in in multiple several instances we have seen that. Now we want to really investigate when given a system how do we go about doing this decomposition and we note that there are primarily 2 approaches that have emerged over the last 50, 60 years that software development has been practiced.

First is the algorithmic decomposition approach and the second is the object-oriented decomposition approach. We will take a look into each one of them and then towards the end, kind of compare them to see what has an advantage over the other.

(Refer Slide Time: 04:10)



Algorithmic Decomposition

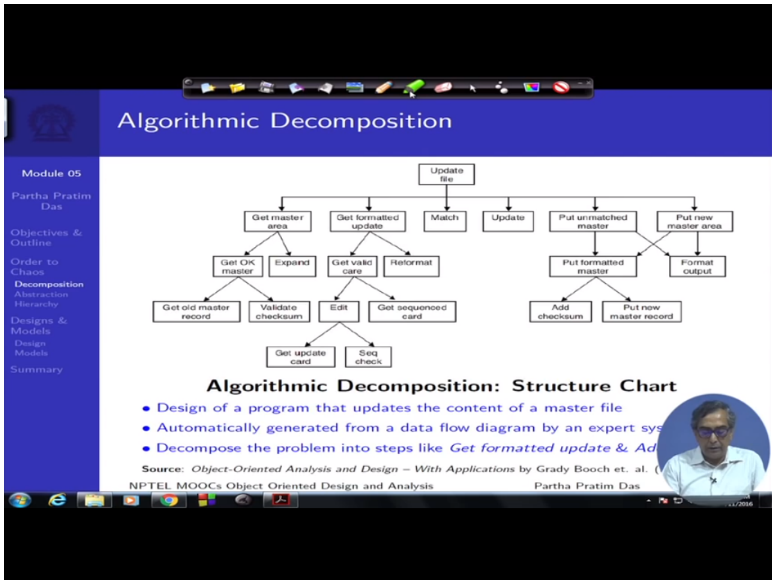
- Top-down structured design
- Divide-and-Conquer
- Decompose the problem into key processing steps

So algorithmic decomposition, this is certainly the most fundamental, all these this has been around for a very long period of time and is primarily a top-down structured approach that given a problem, you can, we can divide it into a number of 2 or more, usually a number of different sub problems which are simpler in nature and whose solutions together can solve the whole problem.

In very specific terms, we have studied emm this kind of approach under divide-conquer kind of algorithmic paradigms while we did binary search and match, sort and things like that. So in general, what we will look at algorithmic decomposition as is to decompose the problem in terms of key processing steps so the main thing here to note in algorithmic deompostion is you will decompose the problem in terms of key processing steps.

So what is inherent of algorithmic decomposition is you identify processing then identify the key processing steps and try to decompose the whole problem in terms of it.

(Refer Slide Time: 05:35)



Algorithmic Decomposition

Module 05

Partha Pratim Das

Objectives & Outline

Order to Chaos

Decomposition

Abstraction

Hierarchy

Designs & Models

Design Models

Summary

Algorithmic Decomposition: Structure Chart

- Design of a program that updates the content of a master file
- Automatically generated from a data flow diagram by an expert sys
- Decompose the problem into steps like *Get formatted update & Ad*

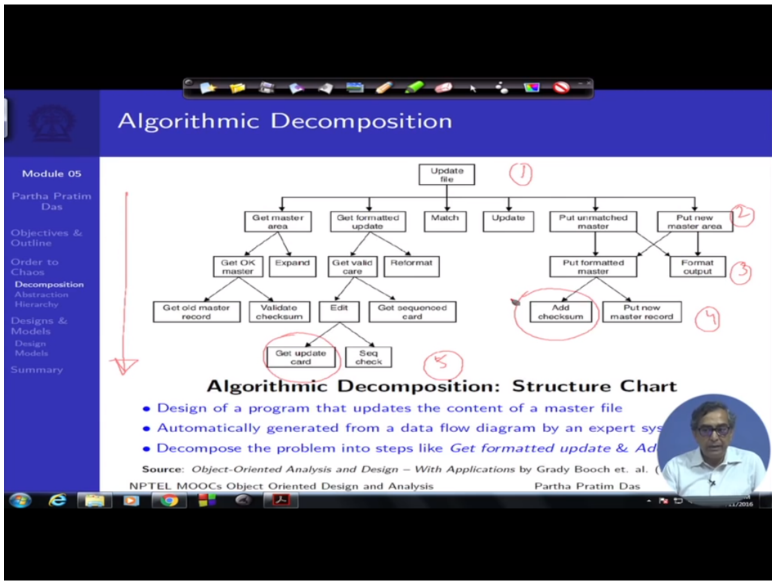
Source: *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (

NPTEL MOOCs Object Oriented Design and Analysis Partha Pratim Das

So here is an example. This example is is from our text book so here the problem is to update a file. So this is the top level problem that we want to actually address, solve. So for doing that, certainly a file will have a master area where the information about the file is written. Then the format of the file then there are several matching and update operations that need to be done. If I am writing the updating the file, the new master area need to be put up and so on.

So based on that the update file, this whole task of update file is divided in terms of a number of subtasks or sub goals which are given in this layer. So this is one level of decomposition where this is the first level in which the tasks are decomposed. Then if you concentrate on any any specific of this tasks they get master area this task has to accomplished, then this can be done in terms of going to other subtasks of get the ok master and to expand.

(Refer Slide Time: 06:00)



Algorithmic Decomposition

Algorithmic Decomposition: Structure Chart

- Design of a program that updates the content of a master file
- Automatically generated from a data flow diagram by an expert sys
- Decompose the problem into steps like *Get formatted update & Ad*

Source: *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (

NPTEL MOOCs Object Oriented Design and Analysis   Partha Pratim Das

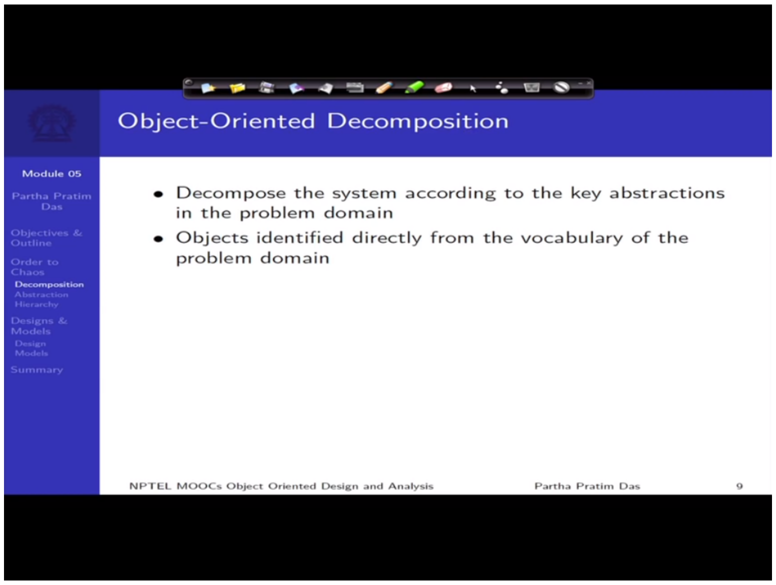
And really does not matter that much if you do not specifically understand every detail of what these subtasks are in this particular case because what we are trying to explain is a broad idea how decomposition can be done. So you can see that over this whole tree of this is the root level and 2,3,4 and 5 levels, we try to solve this problem by algorithmic decomposition and whenever we reach a level say get update card, add checksum, task which do not have other sub tasks.

That would mean that is at a level where it can be directly implemented in terms of an algorithm possibly in terms of a specific function. So this is one way to um actually try to decompose a problem and start solving it and this particular kind of a chart is known as a structure chart and it shows how the tasks are structured and in this particular case this chart was generated from an expert system tool.

So that is interesting because that means that a quite a bit of automation can be done on this. The input was a data flow diagram. You may not; right now you are aware of what a data flow diagrams are over a course of these discussions, we will also talk about data flow diagrams. But an expert system tool can take the data flow diagram which basically says how data is going through the system and make this structure chart of algorithmic decomposition

So in this it has divided the problem into very specific different steps like get formatted update as an here or add checksum as an here or some of them are primitive tasks and some are still composite tasks which need further subdivision. So this is the basic approach of decomposition you look at the whole task. We are basically going by the processing logic, try to divide them into subtasks, take up the subtask and try to divide that further into smaller subtasks till you get into primitive tasks that can be directly solved.

(Refer Slide Time: 09:12)



Object-Oriented Decomposition

- Decompose the system according to the key abstractions in the problem domain
- Objects identified directly from the vocabulary of the problem domain

Module 05

Partha Pratim Das

Objectives & Outline

Order to Chaos

Decomposition

Abstraction

Hierarchy

Designs & Models

Design Models

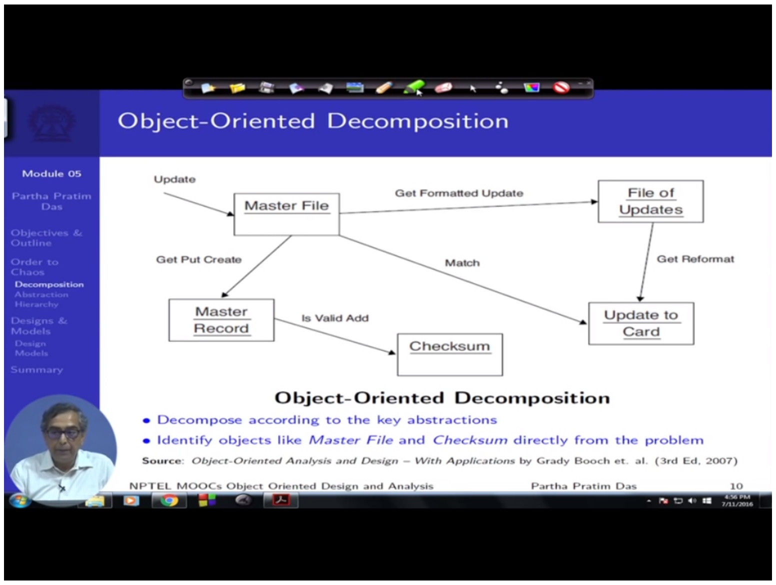
Summary

NPTEL MOOCs Object Oriented Design and Analysis Partha Pratim Das 9

In contrast, in contrast we can do an object-oriented decomposition. In object-oriented decomposition, we do not look at the specific tasks. Rather we try to decompose the system according to the key abstractions of the problem domain that is we take a look into directly the vocabulary of the problem domain. What is the problem domain talking about? Updating file means what are the basic concepts, what are the basic entities, the basic agents that the problem is talking about.

And we will try to decompose the system in terms of those key concepts, key ideas which we referred to here as key abstractions and then try to model and solve the problem. So it is it is quite different from the way the algorithmic approach will go.

(Refer Slide Time: 10:10)



Object-Oriented Decomposition

Module 05

Partha Pratim Das

Objectives & Outline

Order to Chaos

Decomposition

Abstraction

Hierarchy

Designs & Models

Design Models

Summary

Object-Oriented Decomposition

- Decompose according to the key abstractions
- Identify objects like *Master File* and *Checksum* directly from the problem

Source: *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007)

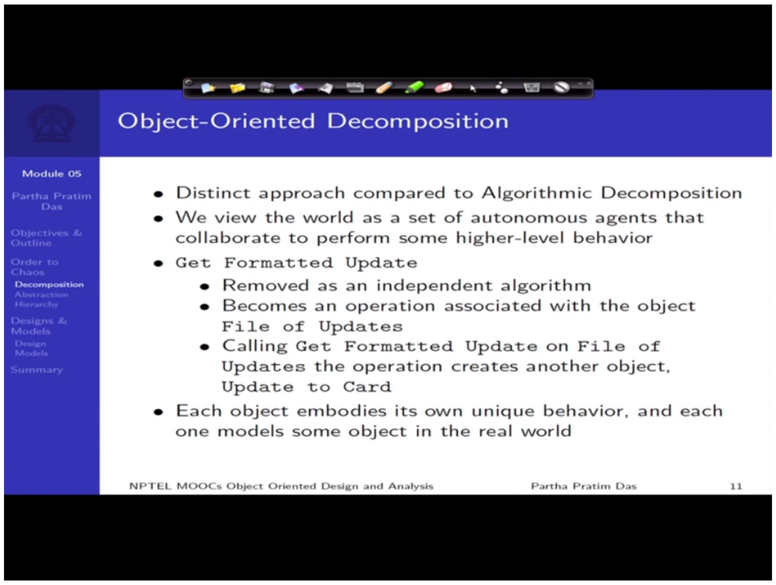
NPTEL MOOCs Object Oriented Design and Analysis Partha Pratim Das 10

So in here the same problem of update representation, here we basically divide the problem into number of different agents. One we say is master file, naturally that needs to keep track of what is happening to the other files. This is the file of updates, this is update to card, the checksum and so on. Each one of them is a key concept, is a key abstraction in the whole problem domain and if you look into the arrows,

These arrows mean that file of updates is an object which support this operation that is master file can invoke gets formatted update on this file of update objects or master file can invoke get put or create operations on the master record object. So the view has become completely different. There is there is no specific get formatted update functionality and the breakdown of that, rather gets distributed in terms of a number of different objects

And those objects are identified objects like these identified directly from the problem domain where we find that these are terms that these are the vocabulary that the problem domain dealing with.

(Refer Slide Time: 11:50)



Object-Oriented Decomposition

Module 05

Partha Pratim Das

Objectives & Outline

Order to Chaos

Decomposition

Abstraction

Hierarchy

Designs & Models

Design Models

Summary

- Distinct approach compared to Algorithmic Decomposition
- We view the world as a set of autonomous agents that collaborate to perform some higher-level behavior
- `Get Formatted Update`
  - Removed as an independent algorithm
  - Becomes an operation associated with the object `File of Updates`
  - Calling `Get Formatted Update` on `File of Updates` the operation creates another object, `Update to Card`
- Each object embodies its own unique behavior, and each one models some object in the real world

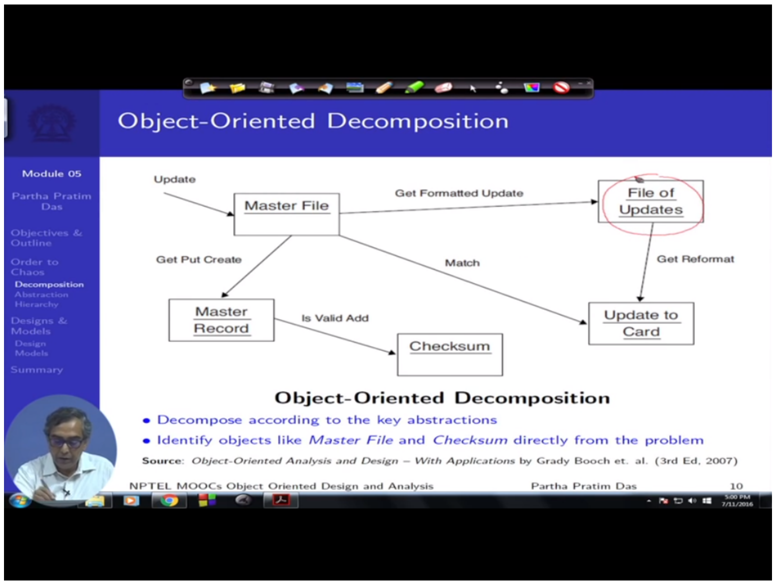
NPTEL MOOCs Object Oriented Design and Analysis Partha Pratim Das 11

So this object-oriented decomposition approach is quite distinct compared to the algorithmic decomposition approach. The first thing is we view the whole world as a set of autonomous agents. This statement is a simple short statement but this is a very strong statement about object-oriented decomposition in fact about object-oriented analysis and design. That is why we think of this whole, whole system as if it is a collection of certain autonomous agents.

autonomous means active entities which can behave on their own kind of if we if we close our eyes and think about the system as if there are a number of agents floating around in the in the universe and each one of them can independently act based on what its behavior is, based on what a it is requested by others to do and based on what it needs to request others to do. So we will take a look into those aspects but just remember this is the basic view

That is a number of autonomous agents collaborate, work together to perform some high level behavior. So let us say how. How does our original functionality of get formatted update how does that happen. This is now removed as an algorithm so we dint have we do not have any specific word among this. It now has become an operation with the object file of updates.

(Refer Slide Time: 13:47)



Object-Oriented Decomposition

Object-Oriented Decomposition

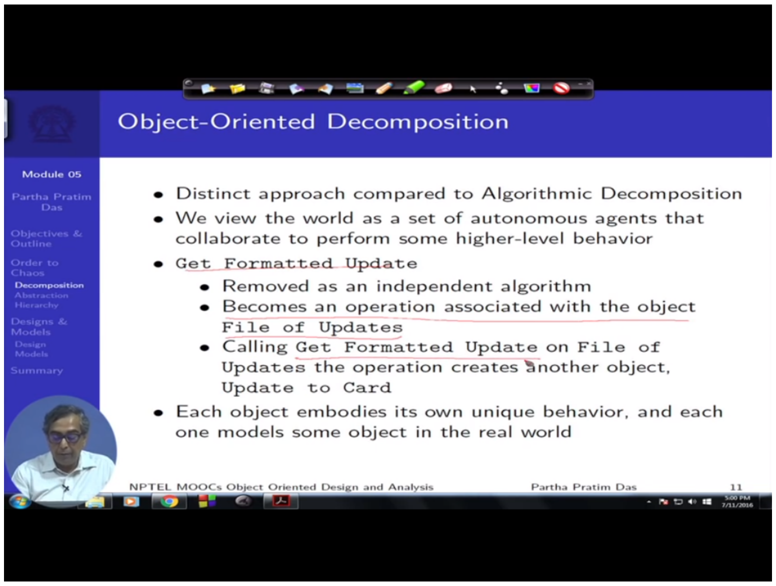
- Decompose according to the key abstractions
- Identify objects like *Master File* and *Checksum* directly from the problem

Source: *Object-Oriented Analysis and Design – With Applications* by Grady Booch et. al. (3rd Ed, 2007)

NPTEL MOOCs Object Oriented Design and Analysis Partha Pratim Das 10

If you are unclear about this let us just one step go back and if you if you see here now this file of updates, this object has this as a method. This as a request that master file can put to this particular object. So this is the difference, this is the change in structure that is happening between the algorithmic and the object-oriented decomposition.

(Refer Slide Time: 14:14)



Object-Oriented Decomposition

- Distinct approach compared to Algorithmic Decomposition
- We view the world as a set of autonomous agents that collaborate to perform some higher-level behavior
- `Get Formatted Update`
  - Removed as an independent algorithm
  - Becomes an operation associated with the object `File of Updates`
  - Calling `Get Formatted Update` on `File of Updates` the operation creates another object, `Update to Card`
- Each object embodies its own unique behavior, and each one models some object in the real world

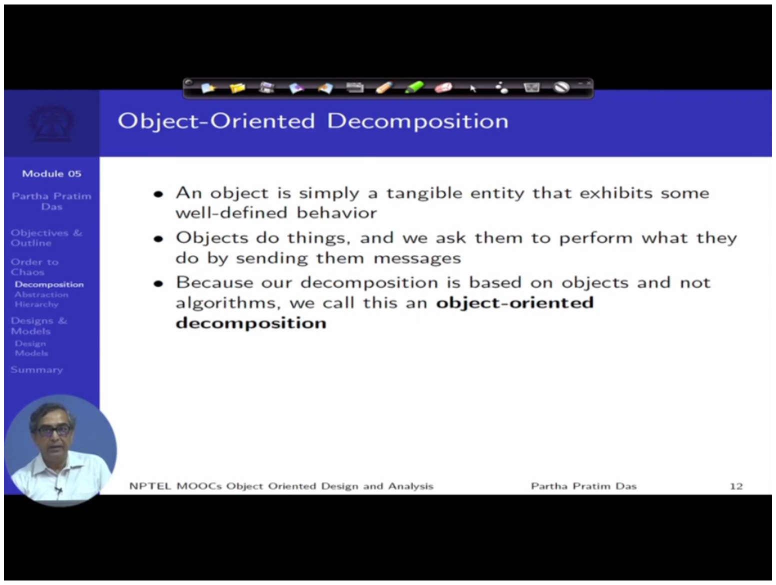
NPTEL MOOCs Object Oriented Design and Analysis Partha Pratim Das 11

So that is why we are saying that now the get formatted update becomes an operation associated with this object. Similarly when we call this get formatted object on the file of updates, this is an object, the operation creates another object update to card. So it will in turn create an object which is a update to card. As I show you in that diagram, let us say, here ah as ah this is operation is invoked on this object, this in turn invokes a get reformat which generates a new update to card object.

So the way the whole thing is modeled, the way the whole thing is now being decomposed and represented are very different and all are in terms of this concept of autonomous agents which are now this particular file of update object, update to card object, these are collaborating to actually achieve the functionality that earlier which are represented in terms of a complete tasks, subtasks decomposition for algorithmic approach.

So every object has its own unique behavior and each one models something in the real world. Now this is this is this is not easy to see ah at this stage one. You you may be feeling confused. As you do more and more examples this will become clear. But keep this in mind that these autonomous agents are not arbitrary. Every autonomous agent must correspond to some basic concept, some basic unit behavior in the real world, in the problem domain that you have. Only then your design would become really effective.

(Refer Slide Time: 16:18)

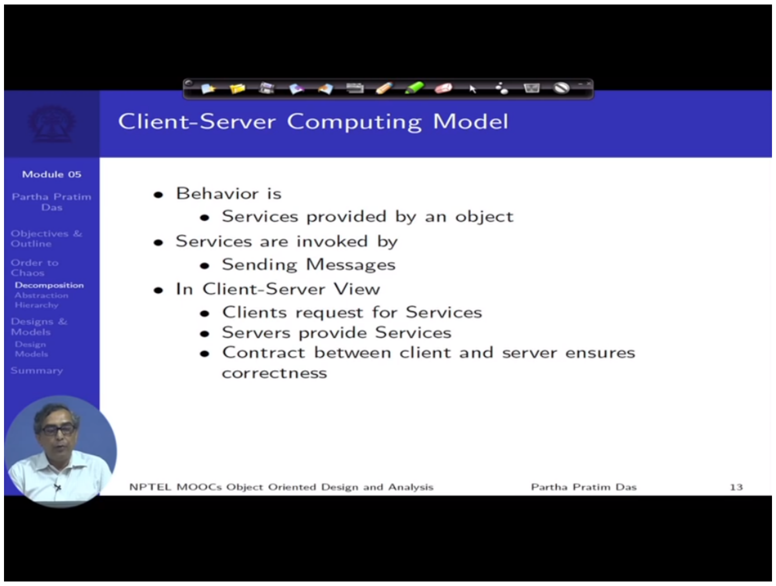


Object-Oriented Decomposition

- An object is simply a tangible entity that exhibits some well-defined behavior
- Objects do things, and we ask them to perform what they do by sending them messages
- Because our decomposition is based on objects and not algorithms, we call this an **object-oriented decomposition**

So an object is simply a tangible entity that exhibits some well-defined behavior. Tangible in the sense that in the real world it has some proper existence and it can do things that we ask them to perform and they do this by, this is interesting they do this by sending them messages that is the main thing that the objects can do they can exist on their own as autonomous agents, they can perform some tasks which we say are operations

And they can send some information, some message to other objects to perform some tasks. This is the whole view and since this whole view is based on the objects rather the autonomous agents and not on the specific algorithms that they perform we call this a object-oriented decomposition. (Refer Slide Time: 17:10)



Client-Server Computing Model

Module 05

Partha Pratim Das

Objectives & Outline

Order to Chaos

Decomposition

Abstraction

Hierarchy

Designs & Models

Design Models

Summary

- Behavior is
  - Services provided by an object
- Services are invoked by
  - Sending Messages
- In Client-Server View
  - Clients request for Services
  - Servers provide Services
  - Contract between client and server ensures correctness

NPTEL MOOCs Object Oriented Design and Analysis Partha Pratim Das 13

So let us take a look at them if if this is how we think about doing our decomposition then how is our problem getting solved. If have done an algorithmic decomposition then we all know how the problem get solved. We all know how what is the computing model. What is a computing model? Algorithms correspond to basically functions in in some high level languages as you can perceive

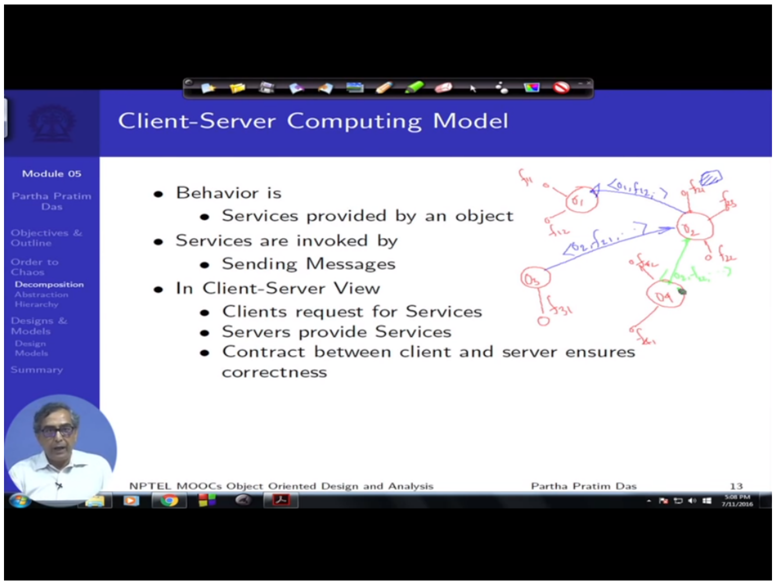
And we always know that functions can be executed in the CPU crossly in a one and one model any algorithm can be executed or if I talk in abstract terms then using tuning machines, the algorithms can be executed so we know that it is it always have data in the memory. It will have instructions or operations or primitive tasks that it can do in terms of the instructions. It will take the data, perform the instruction and put the data back. That is the basic computing model.

In a object based system, the computing basic the idea of basic computing model itself is very different and this is because of 2 reasons. One reason is that when we are talking about object-oriented decomposition which has resulted or object-oriented design has resulted after lot of maturity of software system design. This is not as primitive as a algorithmic decomposition or algorithm based design of systems.

Though by the time we started doing object-oriented design and object-oriented decomposition, we are equipped with a variety of different computing models, some of them are purely (()) (18:50) some of them are follow certain languages like Fortran accept languages like Fortran, Pascal, C, C++ and so on. While others could have multiple processors, they could have different shared memory models, they could do some remote procedure execution and so on a variety of different computing models over the years have emerged.

So the other aspect of object orientation decomposition is also to keep in mind that I decompose what is my computing model and when I decompose I want to make minimum assumption or rather I should say a minimal assumption about my computing model. I do not want to, I yet do not know what will be the best way to design and implement this system. So I do not want to assume that I will run it on a Pentium 3 or I will implement it on a decal processor with 8 cores and so on, I do not want to make such assumptions.

(Refer Slide Time: 20:02)



Client-Server Computing Model

- Behavior is
  - Services provided by an object
- Services are invoked by
  - Sending Messages
- In Client-Server View
  - Clients request for Services
  - Servers provide Services
  - Contract between client and server ensures correctness

Module 05

Partha Pratim Das

Objectives & Outline

Order to Chaos

Decomposition

Abstraction

Hierarchy

Designs & Models

Design Models

Summary

NPTEL MOOCs Object Oriented Design and Analysis   Partha Pratim Das   13

So I make a simple assumption that it is it is a client server computing model which say that I have a collection of autonomous agents the objects, so object 1, object 2, object 3, object 4 and so on. And on every object I have some possible operations that I can do. So this is known as the services these operations are known as the services offered by the respective objects. So these are the different services that object of.

Now the way you it works is any object can send message to another object. This object say if we just turn over to loop, this mess object can send a message to this object saying that I want to use your method or your operation to 1. So it kind of sends a message like I want to go to o2 and use f21 and so on these are the data on which this f21 should be used and so on. In so what happens is when o2 saves that whatever is this method operation, that method operation will start being performed.

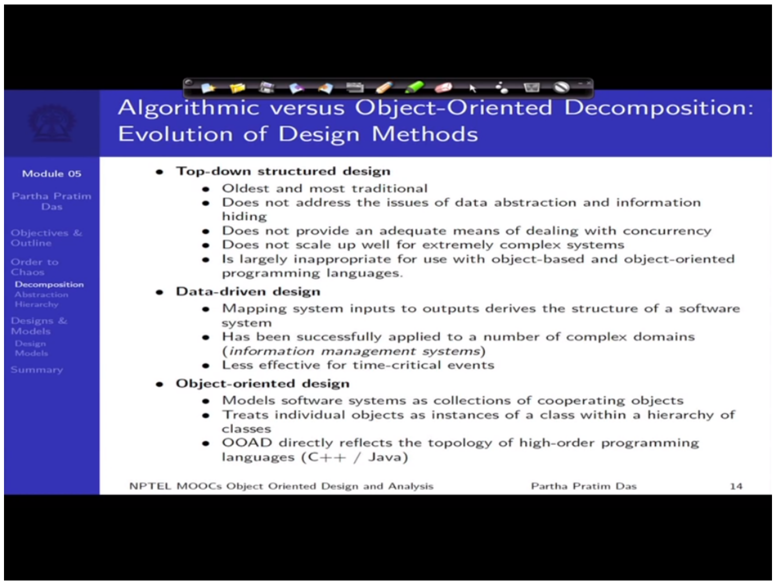
When that is being performed, o2 may need that I would need the help of the operation f12 of object o1 so it sends a message to object o1 with f12 with associated data. And while this is all these are happening simultaneously on a separate note, o4 may have sent another request to o2 for doing some operation f2 and all these can keep on happening at the own speed case at the own speed independence and this is what is known as the client server model.

Why this is anybody who requests for example here if o3 was requesting so we will say here is the client and who was being requested, o2, this is called the server. So as if as as you say that we we say that ah ask for services for the human clien vendors, it is the similar concept that the client will ask for the service from an object. Now there is no specific object as a client, not a specific object as a server, it depends on who is in that what mode.

In a whenever a message request is sent the sender who is requesting for the services is the client and the receiver who will provide the service is a server and this happens in a completely distributed manner. So you can say understand that as long as you you can have some computing system some realizable system, some implementable system where you can have this autonomous agents or objects and you have some mechanism to pass messages, send messages you will be able to implement a client server model.

So if any concrete terms we talk about say C++, then basically objects are instances of classes and the sending of messages is nothing but nothing other than actually invoking or calling method functions. So that is the basic model like just little bit discussed it so that we go forward and keep on doing more and more of how the complex systems are designed, we will have this particular model in the background.

(Refer Slide Time: 24:00)



Algorithmic versus Object-Oriented Decomposition: Evolution of Design Methods

Module 05

Partha Pratim Das

Objectives & Outline

Order to Chaos

Decomposition

Abstraction

Hierarchy

Designs & Models

Design

Models

Summary

- **Top-down structured design**
  - Oldest and most traditional
  - Does not address the issues of data abstraction and information hiding
  - Does not provide an adequate means of dealing with concurrency
  - Does not scale up well for extremely complex systems
  - Is largely inappropriate for use with object-based and object-oriented programming languages.
- **Data-driven design**
  - Mapping system inputs to outputs derives the structure of a software system
  - Has been successfully applied to a number of complex domains (*information management systems*)
  - Less effective for time-critical events
- **Object-oriented design**
  - Models software systems as collections of cooperating objects
  - Treats individual objects as instances of a class within a hierarchy of classes
  - OOAD directly reflects the topology of high-order programming languages (C++ / Java)

NPTEL MOOCs Object Oriented Design and Analysis Partha Pratim Das 14

So we have discussed the basic of algorithmic design, algorithmic decomposition, and the object-oriented decomposition and we will next take a look into how do they compare based on that historical evolution.