

Software Engineering
Prof. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 03
Introduction- III

Welcome. In the last lecture we were discussing about the human cognitive limitations. And we said that while developing large software this is a main problem that is faced. So, software engineering which is essential for developing large software tries to restrict this complexity so that the effort and time required to develop software is linear with the size of the software. Otherwise, if the software engineering principles are not applied and a intuitive approach like a build and fix module is used then the complexity growth; and that is basically the translates into the cost, effort, and the time for development increases exponentially.



And then we were discussing about how does software engineering principles tackle this complexity. And we had said that there are two major principles involved: one is abstraction and the other is decomposition.

In this lecture let us try to understand these two techniques well, because we will see that in almost every technique that is used by software engineering these two are the basic principle behind this. So, let us discuss this abstraction and decomposition techniques.

(Refer Slide Time: 02:12)

What is Abstraction?

- Simplify a problem by omitting unnecessary details.
 - **Focus attention on only one aspect of the problem and ignore other aspects and irrelevant details.**
 - Also called model building.

 IIT KHARAGPUR |  NPTEL ONLINE CERTIFICATION COURSES

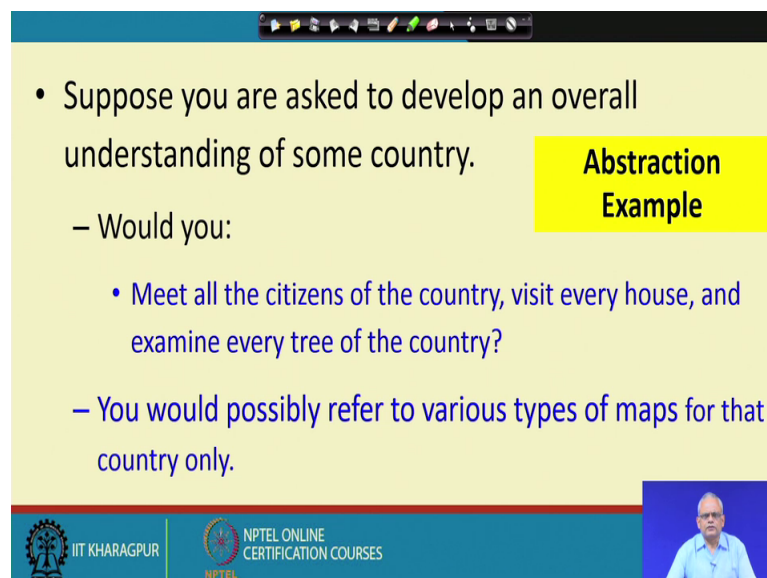
29

First thing is: what do you mean by abstraction? Abstraction is basically leaving out unnecessary parts and focusing on some parts that we require. Given a large problem if we look at the problem with entirety, then it appears very complex. We would like to focus only one aspect of the problem and ignore the rest. So, that is basically called as Obstruction. So, we focus our attention only one aspect of the problem and ignore other aspects that are irrelevant to the point that we are focusing and this is also called as Model Building.

Here we simplify a problem by omitting the unnecessary details and that is basically constructing a model. Every abstraction requires construction of a model and a model focuses on one aspect, and it ignores rest. For example: we have a very large and complex building to be developed and we just want to focus that how will it appear. Then we will construct a frontal model of a building, we ignore the rest what is its material it is build off, what is the thickness what is the floor plan, internal design, and so on we just focus on the frontal view of the building.

So to summarize this abstraction, every abstraction require some model building and a model basically focuses on some aspect and ignores the rest.

(Refer Slide Time: 04:23)



- Suppose you are asked to develop an overall understanding of some country.

Abstraction Example

- Would you:
 - Meet all the citizens of the country, visit every house, and examine every tree of the country?
- You would possibly refer to various types of maps for that country only.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, let us see some examples of how abstraction can help. Let us look at a hypothetical case that you have been asked to develop an overall understanding of some country. Maybe, you be later given the responsibility to rate the marketing department of some


company for a country let us say, and you have been asked that please develop a overall understanding of a country.

So, to do this what would you do? Would you go to that country, move around try to meet people, look at where are the mountains, where the rivers and so on? If you really did that then it will be extremely complex task, it will take you tens or hundreds of year still you will not be able to develop a proper understanding. You will not like to meet all the people examine every tree in the country, river, mountain and so on. But what you would really do is refer to various types of maps of the country.



(Refer Slide Time: 05:51)


You would study an Abstraction...

- A map is:
 - An abstract representation of a country.
 - Various types of maps (abstractions) possible.



The slide displays two maps of India side-by-side. The left map is a political map showing state boundaries and major cities. The right map is a physical map showing topographical features like mountains, rivers, and the Bay of Bengal. A small inset map of India is also visible in the top right corner of the slide content.

 IIT KHARAGPUR |  NPTEL ONLINE CERTIFICATION COURSES



The maps are basically model or an abstraction of a country. You will study the political map, that focuses on the different provinces, capital, major cities, railway road connectivity and so on. You will look at the physical map: we will try to find out the vegetation, the elevation of the different places, rivers, sea shore and so on.

So, an abstraction can help solve the complex problem very easily. And one thing to notice is that there are various types of abstraction for the same problem.

(Refer Slide Time: 06:48)

Does every Problem have a single Abstraction?

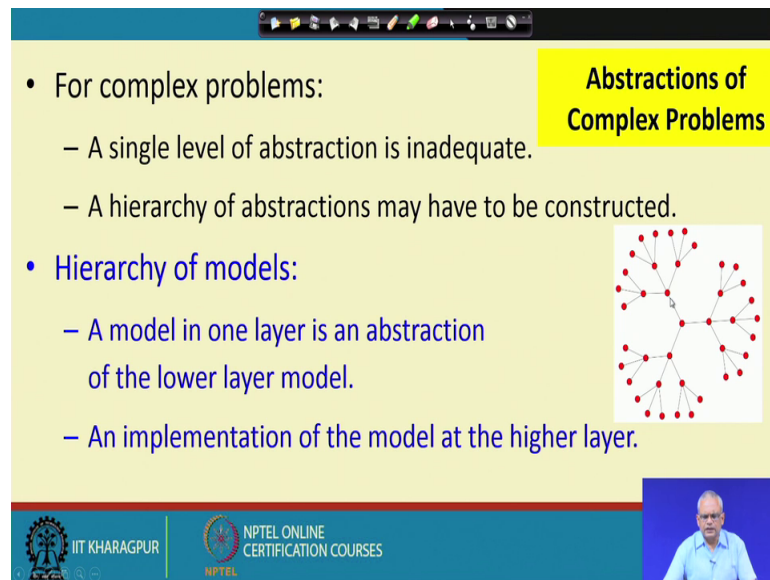
- Several abstractions of the same problem can be created:
 - Focus on some specific aspect and ignore the rest.
 - Different types of models help understand different aspects of the problem.

The slide features two maps of India: one showing a topographic view and another showing a political boundary view. At the bottom, there are logos for IIT Kharagpur and NPTEL Online Certification Courses, along with a small video inset of a speaker.

But then, is it that for every problem we can be happy with just one abstraction. Can just one abstraction help us develop a full understanding of a problem? No.

Several abstraction to the same problem can be created. These focus on some specific aspect and ignore the rest and the different types of models help to understand different aspects of the problem. For example, in the case of a building we would like a frontal model, we like to develop a prototype, we like to develop a floor plan, we like to develop a thermal model of the building and so on. So, for the same problem various models can be created each focusing on some aspect of the problem.

(Refer Slide Time: 07:54)



Abstractions of Complex Problems

- For complex problems:
 - A single level of abstraction is inadequate.
 - A hierarchy of abstractions may have to be constructed.
- Hierarchy of models:
 - A model in one layer is an abstraction of the lower layer model.
 - An implementation of the model at the higher layer.

The slide features a yellow background with a blue header and footer. A yellow box in the top right corner contains the title 'Abstractions of Complex Problems'. A tree diagram with red nodes and black lines is positioned on the right side of the slide. The footer includes the IIT Kharagpur logo and the NPTEL Online Certification Courses logo. A small video inset of a speaker is visible in the bottom right corner.

But, if the problem is extremely complex a single level of abstraction may itself be again complicated. We would like to create a hierarchy of obstruction, so if these are the problems we will create a first level abstraction, then we will create a second level abstraction, then an abstraction of those second level abstraction and so on. And if we look at the root level that will be the simplest representation. And as we understand the simplest representation we might look at the next level of the models and so on.

So, for very complex problems we would create a hierarchy of obstructions. As we proceed with software design we will look at hierarchy of obstructions that are used, and in this hierarchical model a model at a any level is actually the details on which the next level of model is built. So, a model is an abstraction of a lower level model and we say that it is a implementation of the higher level model.

(Refer Slide Time: 09:32)

Abstraction of Complex Problems -- An Example

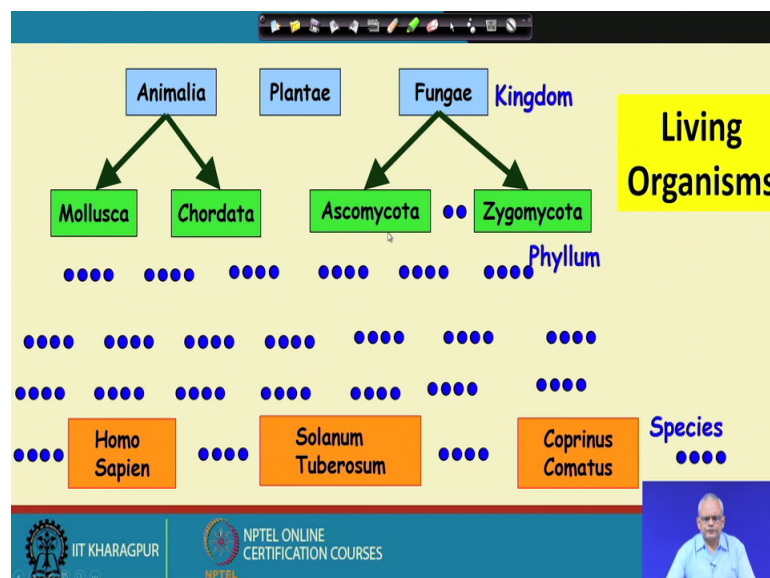
- Suppose you are asked to understand all life forms that inhabit the earth.
- Would you start examining each living organism?
 - You will almost never complete it.
 - Also, get thoroughly confused.
- **Solution: Try to build an abstraction hierarchy.**

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Let us look at abstraction of a complex problem; how a hierarchy can help, a hierarchical set up models can help. Let us look at a very complex problem that you have been asked to understand all the life forms that inhabit the earth. And remember there are billions or trillions of species.

So, in order to understand this life form if you go on examining various species nobody can complete the study in his lifetime. But then, what you do is you create an hierarchy of abstraction.

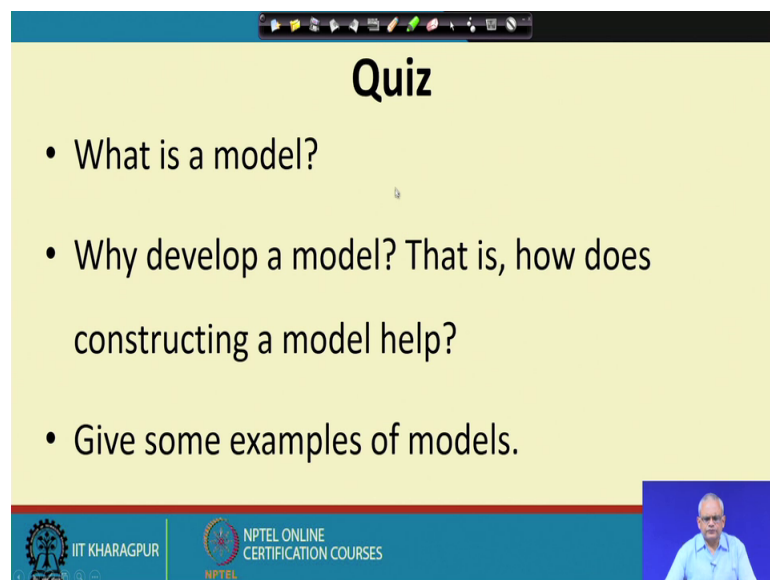
(Refer Slide Time: 10:27)



If you look at a biology book you will find such an abstraction, because they want to explain the reader the simplest way and they have constructed this abstraction that at the top level there are only three types of living organisms: the animals, plant, and the fungus. And then we have the mollusca, chordate, etcetera, etcetera. And there are further hierarchies until you reach the bottom most. And these are the species I was just saying that there will be trillions or species here.

So, for a complex problem the number of layers in the hierarchy can be large and each of these layers is basically a model.

(Refer Slide Time: 11:29)



Quiz

- What is a model?
- Why develop a model? That is, how does constructing a model help?
- Give some examples of models.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

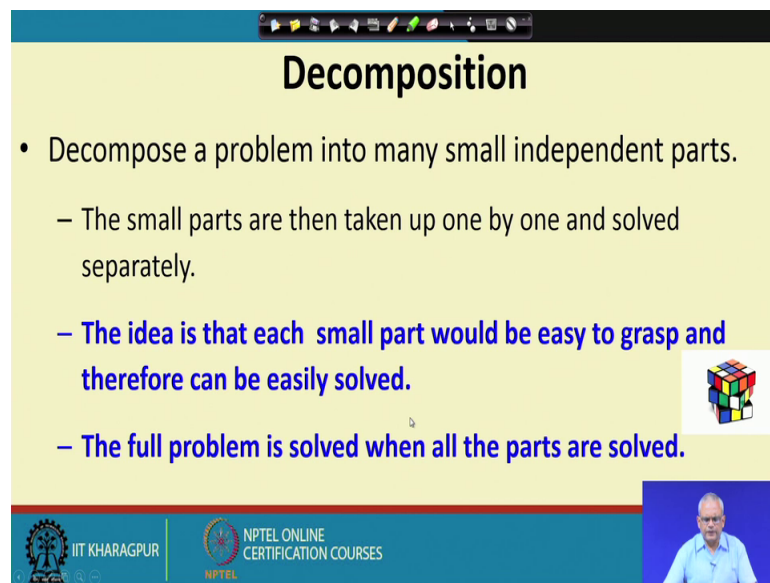
Let us just try to examine our understanding of what we discussed, because this is after all a very important principle what exactly is a model. We said that a model is an abstraction, where we focus on some issues and we ignore the rest and several types of models can be there for a problem. And if it is a complex problem we would like to create a hierarchy of models.

But then the next question that we would like to answer is that why is it necessary to develop a model, how does construction of a model help. The answer is straight forward that understanding a complex problem is extremely difficult due to human cognitive limitations. We would like to first understand the simplest form of the problem. That said the top of the hierarchy and then we will look at the other models to develop an incremental understanding of the problem.

The third question is give some examples of models that you are aware off, try to think the different items that you deal with everyday and can you think of places where model is used. What about the content page of a book, is it a model of the book, what about a simulation experiment? Are we constructing a model and then trying to experiment on the model as if we are experimenting on the real system and so on.

So, you may come up with hundreds of different models that we use in the real life.

(Refer Slide Time: 13:47)



Decomposition

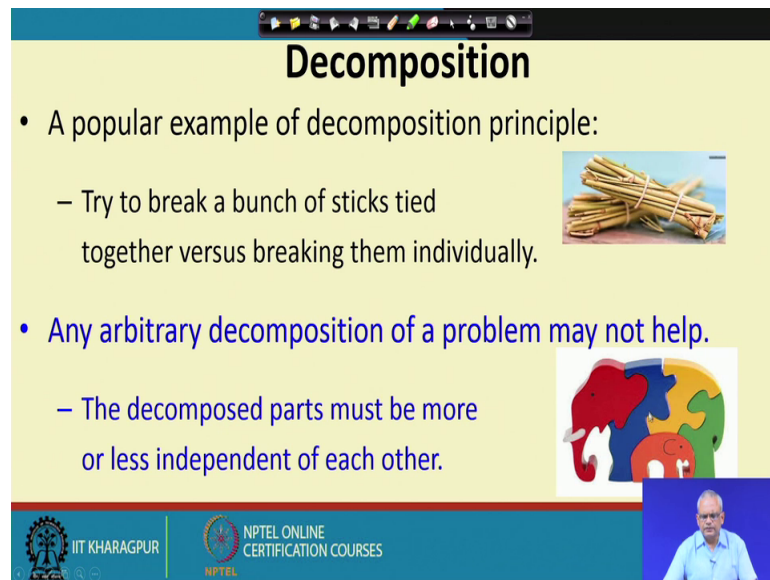
- Decompose a problem into many small independent parts.
 - The small parts are then taken up one by one and solved separately.
 - **The idea is that each small part would be easy to grasp and therefore can be easily solved.**
 - **The full problem is solved when all the parts are solved.**

The slide includes a small image of a Rubik's cube and a video inset of a speaker in the bottom right corner. The footer contains the logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES.

Now, let us look at the other important principle which is decomposition. Decomposition as it means is to decompose a complex problem into small independent parts, because the problem in its entirety is extremely complex hard to understand it takes exponentially large time, but we would like to break it into small problems. And then we look at examine and understand the small problems and then once we put them together we have the understanding of the entire problem.

Have you come across some such principles in daily life? Yes, there were many places where we use decomposition. We divide a large problem into small parts, and then these small parts are easily grasped and solved and then we put them together this solution to the small parts and we have the solution to the full problem.

(Refer Slide Time: 15:06)



Decomposition

- A popular example of decomposition principle:
 - Try to break a bunch of sticks tied together versus breaking them individually.
- Any arbitrary decomposition of a problem may not help.
 - The decomposed parts must be more or less independent of each other.

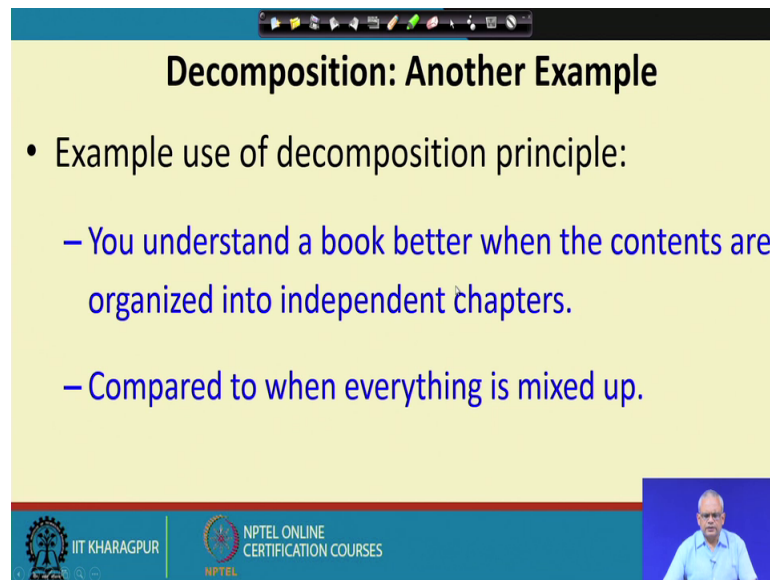
The slide features a presentation control bar at the top. Below the title, there are two main bullet points. The first bullet point is accompanied by an image of a bundle of sticks tied together. The second bullet point is accompanied by an image of a colorful elephant made of puzzle pieces. At the bottom of the slide, there are logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, along with a small video inset of a man in a blue shirt.

One of the very popular example of the decomposition principle is that if you try to break a bunch of sticks tied together it would be extremely complex, but then if you decompose it then you can break the sticks individually.

But then, one what a question when using the decomposition principle is that a arbitrary decomposition of a problem may not help. We have to decompose such that each small problem can be solved separately. If we want to let us say draw an elephant, if you decompose in arbitrary ways it does not help. We thought that we will just draw each of them separately and then we will put them together, but each individual part drawing that may not be much easier.

So, we need to decompose properly; we will see how to properly decompose a problem and then we can solve it easily. And this is also a corner stone in all software engineering principles. Let us look at some examples.

(Refer Slide Time: 16:38)



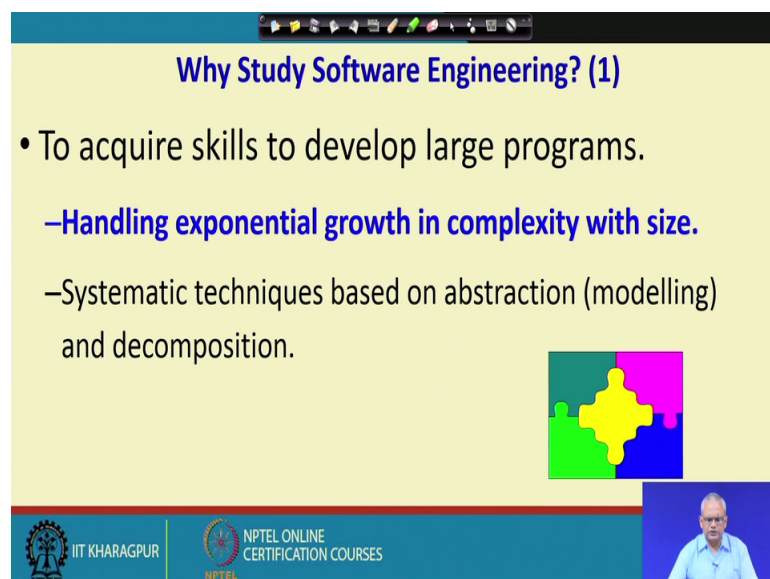
Decomposition: Another Example

- Example use of decomposition principle:
 - You understand a book better when the contents are organized into independent chapters.
 - Compared to when everything is mixed up.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

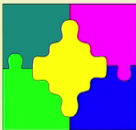
Let us say a book is written half (Refer Time: 16:47) it is all mixed up. It would become extremely complex for somebody trying to read the book. What he somebody get helped, what will really help if that a book is nicely decomposed into different chapters each chapter is independent: introduction chapter one deals with something, chapter two deals with something else and so on. So, a chapter organization of a book is actually a decomposition into a complex large thought into small manageable parts.

(Refer Slide Time: 17:31)



Why Study Software Engineering? (1)

- To acquire skills to develop large programs.
 - **Handling exponential growth in complexity with size.**
 - Systematic techniques based on abstraction (modelling) and decomposition.



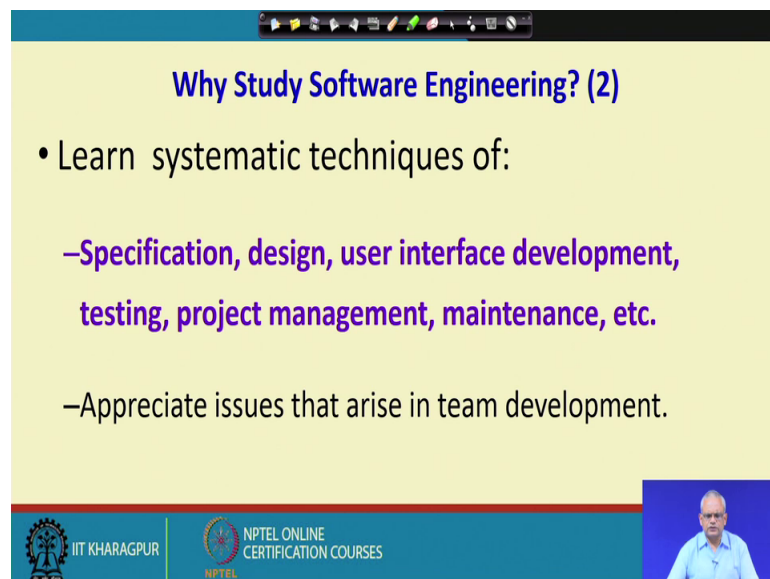
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So far we saw that software engineering really helps when we have large problem to solve as are the industry standard problems. But then, we saw that there are two main techniques: abstraction and decomposition. Now let us see; what are the benefits that we will accrue from study of the software engineering principles.

The first thing is that we will get the skills to develop the large programs, because the intuitive technique of build and fix just try to program and then debug does not work for developing large programs. We should be able to create models that is abstraction, we should be able to decompose the large program, and that would be the way to handle exponential growth in complexity. So, this is a very important skill that how do we use the abstraction and decomposition principles in developing large programs.

As we proceed we will see the techniques that make use of this and help us create models and also decompose a large problem. So, these are systematic techniques which will help us in applying the principles of abstraction and decomposition.

(Refer Slide Time: 19:08)



Why Study Software Engineering? (2)

- Learn systematic techniques of:
 - Specification, design, user interface development, testing, project management, maintenance, etc.
 - Appreciate issues that arise in team development.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

The second most important reason why we need to study software engineering is that we will learn systematic techniques. How to specify a problem, how to go about designing, how to do the user interphase development, how to do testing, how to do project management, maintenance, etcetera. So, these are the various specialized aspects of software engineering and we will get a glimpse of this. And we will appreciate the issues that arise in team based development.

(Refer Slide Time: 19:50)

Why Study Software Engineering? (3)

- To acquire skills to be a better programmer:
 - Higher Productivity
 - Better Quality Programs

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Not only that for large projects software engineering principles will be useful; the skills that we acquire will also help us in developing small programs in a better way, so that even for small programs after studying the software engineering principles we will have a higher productivity and we will be able to write better quality programs.

(Refer Slide Time: 20:18)

Jobs versus Projects

Routine **Uncertainty of outcome**

← ← ← → → →

Jobs **Projects** **Exploration**

Jobs – repetition of very well-defined and well understood tasks with very little uncertainty

Exploration – The outcome is very uncertain, e.g. finding a cure for cancer.

Projects – in the middle! Has challenge as well as routine...

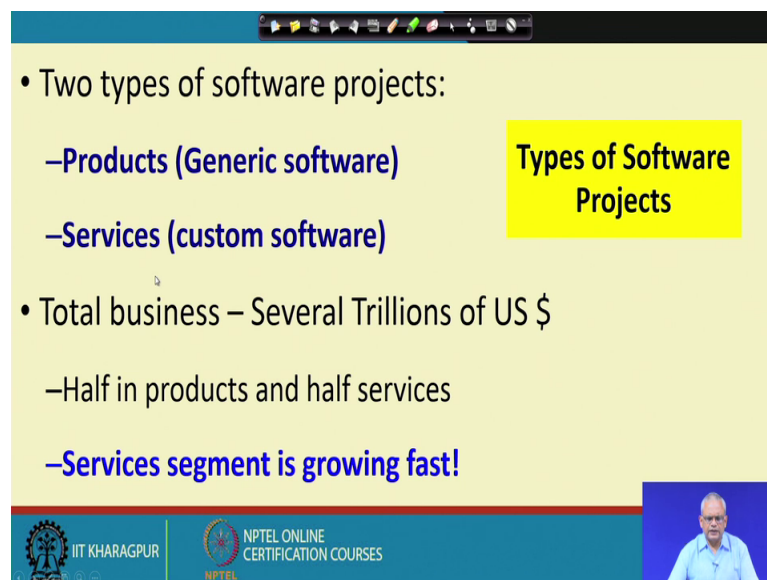
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, let us look at a few basic issues before we look at the software engineering principles. The first thing is that; what is the difference between the job and the project. A job is something which is routine, we can easily do it; project involve some challenge

and exploration is where there is a uncertainty of outcome. A job is a repetition of some well defined and well understood tasks with very little uncertainty. For example: I ask you that go to the market and fetch a chocolate; yes, it can be done you know how to go to the market, you know how to buy a chocolate. So it is a well defined, well understood task can be easily done, there is no uncertainty you can definitely do it.

An exploration is a work whose outcome is uncertain. For example: I ask you that find a cure for cancer. You will go about doing the work, but never know whether there will be any success at all. Project is somewhat in between, between a job and exploration which is completely uncertain and also completely deterministic at the projects where there are many challenges, but then some of the work is also routine.

(Refer Slide Time: 22:08)



The slide, titled "Types of Software Projects", lists the following points:

- Two types of software projects:
 - Products (Generic software)
 - Services (custom software)
- Total business – Several Trillions of US \$
 - Half in products and half services
 - Services segment is growing fast!

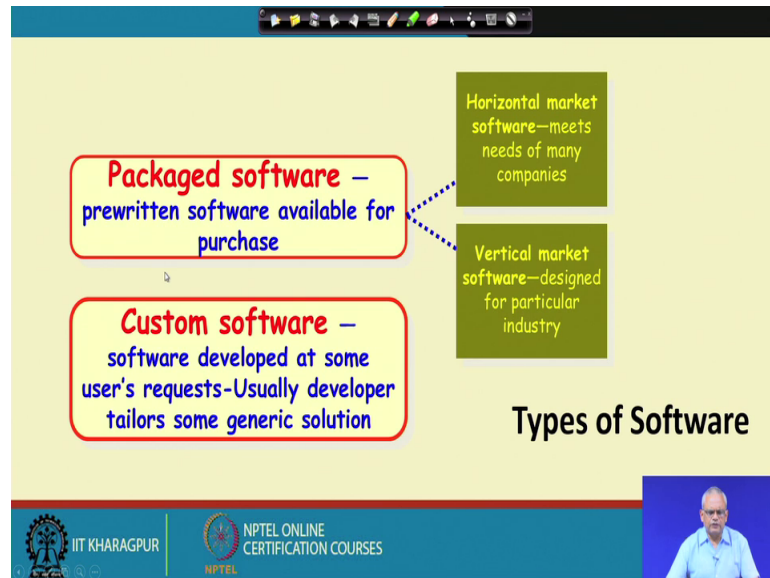
The slide also features the IIT Kharagpur and NPTEL Online Certification Courses logos at the bottom, and a small video inset of a presenter in the bottom right corner.

So, we know that what exactly is a project. A software development project consists of some routine work as well as there are challenges in it. Now let us look at the different types of software projects that currently the industry undertakes.

There are mainly a two types of software projects: one is product development projects or generic software development projects and services projects or custom software development. So, these are the two broad categories of projects: the product development projects and the services. The total size of the industry for software development is extremely large, several trillions of US dollar. And right now nearly half of this is about developing and selling products and the other half is about services.

But then, the growth in services is much more than in products and soon we will have the services market much more than the product market. And just remember that about 50 year back almost everything was product, there was hardly any services.

(Refer Slide Time: 23:58)



Let us examine the issues in the software product development project and the services projects. The product development projects are basically packaged software. These are available of the self. Anybody can order and buy this software. Already this, have been made and you just have to buy. But then, if we look at the packaged software or the software products there are two main types: one is the horizontal product, this meet the needs of many companies. For example: let us say a word processing software or let us say an operating system and so on. These are horizontal, because a large number of companies and individuals are interested.

Whereas, a vertical market software a vertical product focuses only a small group of companies. For example, let us say we may talk about the telecom domain and here there are software for the telecom domain; let us say chemical plant simulation. So, that is another vertical market or the banking software that is another vertical market where only the banks are interested.

So, there are two types of products: one is the horizontal products. This is developed for almost everyone might need this product, whereas the vertical only a specific type of industry they would be interested in the vertical product. But a custom software projects,

these are the services. Here the software is developed at some user's request let us say an academic institution wants to automate its book keeping activities like let us say academic grading, fees collection, establishment, salary, etcetera. So, it might get a software it might request a company to develop its software, but then the company try to develop it will not develop from scratch. It would usually have some solution for educational software and then it will tailor only small parts of it to meet the requirement of this specific educational institute. Every custom software is basically will have some software generic version or let us say done for some customer and then it is tailored or custom made for the other customers.

So, we will just stop here, the time is getting over. We will stop here and we will continue from this point in the next lecture.

Thank you.