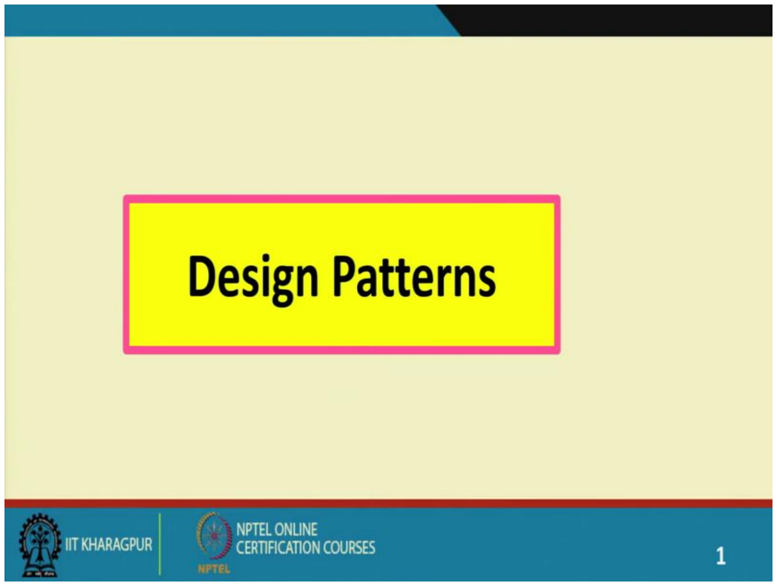
Object-Oriented System Development using UML, JAVA and Patterns

Professor. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur

Lecture 37
Introduction to Design Patterns

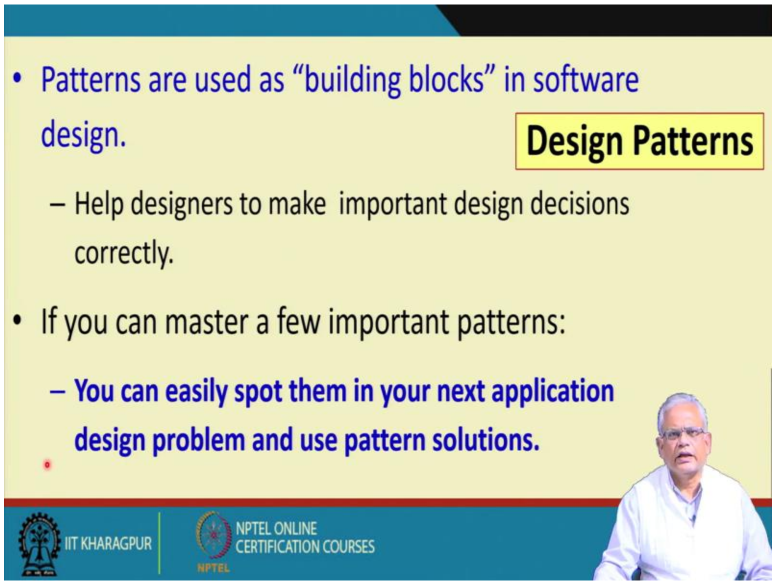
Welcome to this session. In this session, we will start discussing about design patterns. So far, we have looked at the basic design process using which, you get reasonable design and then we had discussed about some principles using which you can get a better design and now we will discuss about the design patterns and we will see that these are very important design elements, good design solutions, which have been worked out by experts who have a lot of expertise in the area of software design and if we know some of these patterns, we can reuse these in our design to get a good design. Let us start discussing about the design patterns.

(Refer Slide Time: 01:15)



Design Patterns

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
1



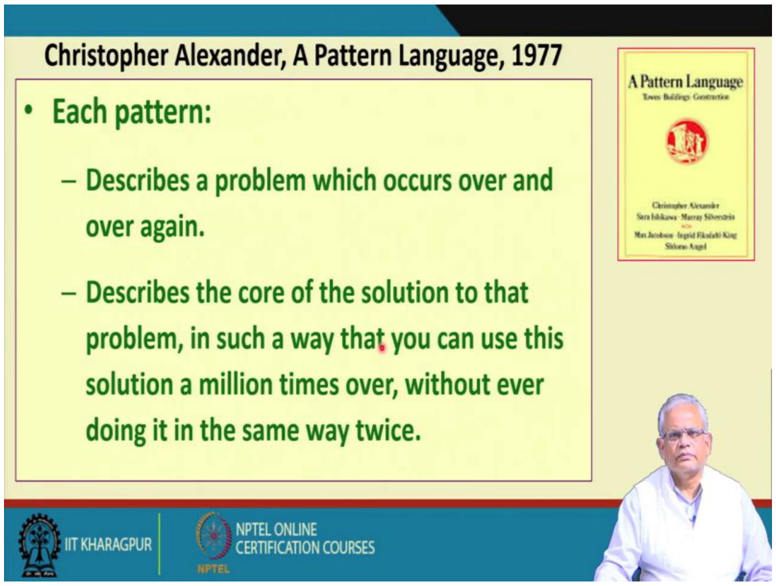
Design Patterns

- Patterns are used as "building blocks" in software design.
  - Help designers to make important design decisions correctly.
- If you can master a few important patterns:
  - **You can easily spot them in your next application design problem and use pattern solutions.**

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

As we were discussing, the design patterns are essentially building blocks. If we know some of these patterns, we can use this building blocks in our design and it helps us to arrive at a correct and good design. We will see that if you can master some of the design patterns, you will be able to spot them next time you do a design, you will be able to spot, where to use these design patterns and you will be able to use this design patterns to arrive at a good design.

(Refer Slide Time: 02:11)



Christopher Alexander, A Pattern Language, 1977

- **Each pattern:**
  - **Describes a problem which occurs over and over again.**
  - **Describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it in the same way twice.**

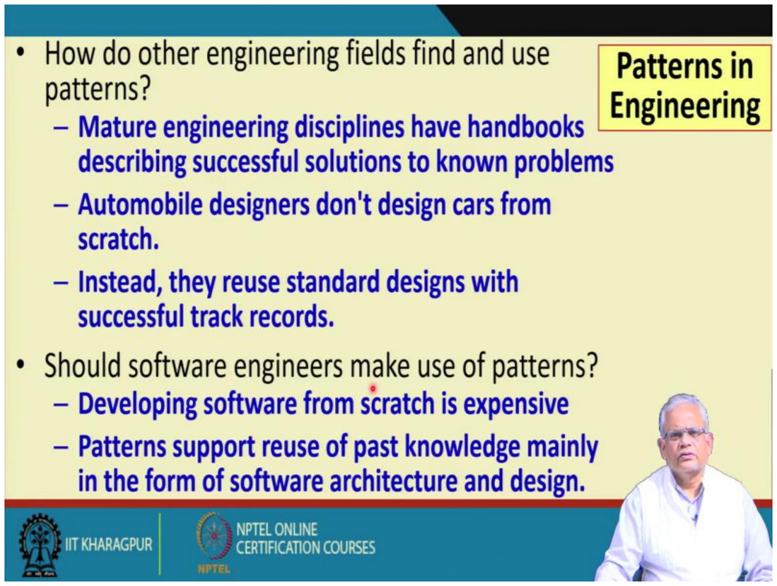
A Pattern Language
Towns Buildings Construction
Christopher Alexander
Sara Ishikawa Murray Silverstein
Max Jacobson Ingrid Fiksdahl-King
Shlomo Angel
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

The design pattern area is inspired by the work of Christopher Alexander, way back in 1977. He was not a computer scientist; he was an architect. But he had got this idea that to design a building there are some reusable solutions which can be identified and then you can have a language to compose these well-known design solutions to arrive at any design. That was the idea he proposed. He wrote this book 'A Pattern Language' which listed more than 300 patterns in designing buildings, architecture of different buildings.

According to him, each pattern describes a problem which occurs over and over again. Whenever we want to build a new building, we will come across some parts which are common to other buildings and if we have the core of the solution described in the form of a book, good solutions described in the form of a book, we can make use of it. But then we do not really plug it in.

He captures that we do not plug in the pattern solution, saying that describes the core solution to the problem in such a way that you can reuse the solution a million times over without ever doing it the same way twice. Which means that the patterns provide a broad solution which we adapt to the specific problem that we have. The design patterns for software is inspired by the work of Christopher Alexander, on patterns for building.

(Refer Slide Time: 04:52)



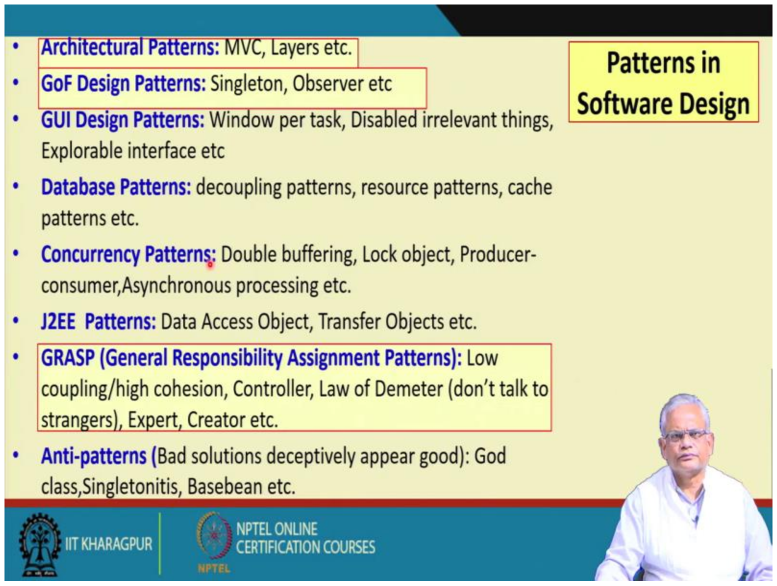
Patterns in Engineering

- How do other engineering fields find and use patterns?
  - Mature engineering disciplines have handbooks describing successful solutions to known problems
  - Automobile designers don't design cars from scratch.
  - Instead, they reuse standard designs with successful track records.
- Should software engineers make use of patterns?
  - Developing software from scratch is expensive
  - Patterns support reuse of past knowledge mainly in the form of software architecture and design.

But if we really think of it, every engineering discipline makes use of some patterns or other even though they do not call it patterns. For example, civil engineering, mechanical engineering, etc., they have handbooks. Even electronics engineering has handbooks of integrated circuits and these integrated circuits for electronics engineers are reusable solutions. And if you know these ICs, the behavior of these ICs, you can plug that in your electronic machine design. The automobile designers do not design a new automobile from scratch. They make use of solutions that have been tried out in other automobiles which have worked out well in other automobiles. But what about software development? Should we use patterns for software development?

It turns out that software has many problems. For example, it is considered to be very expensive, it has a lot of bugs, very difficult to change and many of these arise from the fact that software unlike other engineering disciplines was largely developed from scratch. The design patterns are an important mechanism to codify past knowledge, experience of expert developers and reuse them in our solution. If we know these patterns, what the expert designers had used, we will see that we can reuse and improve the quality of our design.

(Refer Slide Time: 07:10)



Patterns in Software Design

- **Architectural Patterns:** MVC, Layers etc.
- **GoF Design Patterns:** Singleton, Observer etc
- **GUI Design Patterns:** Window per task, Disabled irrelevant things, Explorable interface etc
- **Database Patterns:** decoupling patterns, resource patterns, cache patterns etc.
- **Concurrency Patterns:** Double buffering, Lock object, Producer-consumer,Asynchronous processing etc.
- **J2EE Patterns:** Data Access Object, Transfer Objects etc.
- **GRASP (General Responsibility Assignment Patterns):** Low coupling/high cohesion, Controller, Law of Demeter (don't talk to strangers), Expert, Creator etc.
- **Anti-patterns** (Bad solutions deceptively appear good): God class,Singletonitis, Basebean etc.

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

There are many types of patterns, which are in use for software development. The architectural patterns, these provide the broad framework for development. These do not translate to code actually. But they guide the detailed design, MVC layer, etc., these are architectural patterns. We will look at the MVC pattern as we proceed in our discussion.

The Gang of Four patterns are extremely popular, we will discuss the important Gang of Four patterns: singleton observer etc. And then there are GUI patterns: Windows per task, disabled irrelevant things, explorable interface etc., which we will not discuss. Database patterns: decoupling pattern, resource pattern, cache pattern etc., which we will not discuss. Concurrency patterns: double buffering, lock object, producer consumer, asynchronous processing, etc., which we will also not discuss.

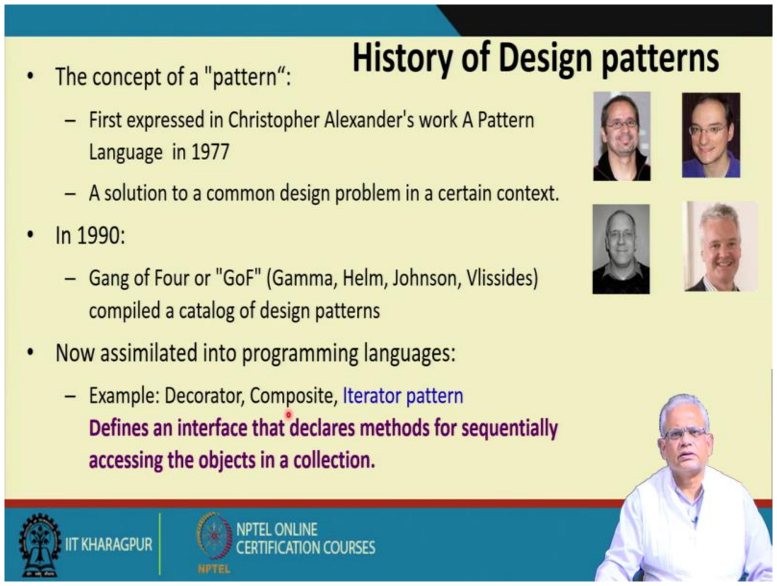
J2EE patterns: data access object, transfer object, etc., we will also not discuss this. The GRASP patterns, this is the General Responsibility Assignment Patterns: low coupling, high cohesion, controller, Law of Demeter, expert, creator, etc., we will discuss this. Turns out to be very simple pattern, many of these appear to be more of common sense and since these are extremely simple, intuitive and these are broad applicability.

Almost every design that you do can be guided by the GRASP patterns and therefore, we will use and discuss the GRASP patterns first. This is extremely simple, intuitive, wide applicability and therefore, we will discuss the GRASP pattern first and then we look at the GoF and the architectural patterns and then, we also have anti-patterns. The anti-patterns are some design solutions which appear to be good but later turn turned out to be liability. Initially when you try to use them, you think that you have done a correct decision, but later will find that it makes the design really bad. There are several examples of anti-patterns, God class, Singletonitis, Basebean, etc., which we will also not discuss considering the limited hours that we have. We will discuss first the GRASP patterns and then we will look at the GoF and we will look at the MVC pattern out of the architectural patterns.

And this other patterns, some of them are specific, if you are doing database development, then you might need the database patterns. Similarly, for concurrency if you are doing concurrent programs, if you are doing J2EE, you are using J2EE, JAVA Enterprise Edition, then you will need these patterns.

Similarly, for GUI development you will need GUI patterns, but these patterns which we are going to discuss the GRASP pattern, GoF pattern and the MVC pattern have wide applicability across different verticals. Once you learn them, you will have enough opportunity to use them in your design development and also if you are trying to understand a solution worked out by somebody you will see that they have made use of these patterns. So, not only in your design you can use, but while understanding somebody's design, you might see that they have made use of these patterns.

(Refer Slide Time: 12:00)



History of Design patterns

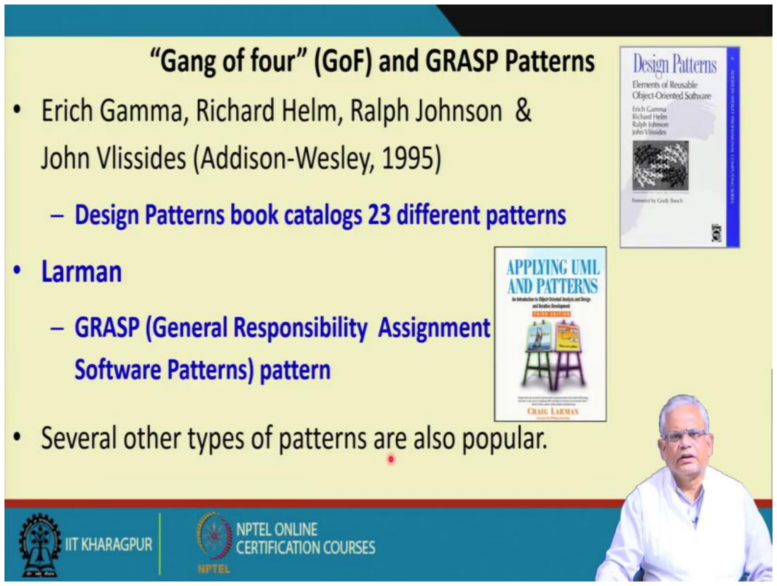
- The concept of a "pattern":
  - First expressed in Christopher Alexander's work A Pattern Language in 1977
  - A solution to a common design problem in a certain context.
- In 1990:
  - Gang of Four or "GoF" (Gamma, Helm, Johnson, Vlissides) compiled a catalog of design patterns
- Now assimilated into programming languages:
  - Example: Decorator, Composite, Iterator pattern
    Defines an interface that declares methods for sequentially accessing the objects in a collection.

We had just remarked that the design patterns originated from the work of Christopher Alexander where he proposed a pattern language for buildings, 1977. These are some good solutions to commonly occurring problems, which work in some context. In 1990s, the Gang of Four: Gamma, Helm, Johnson and Vlissides compiled a catalogue of design patterns for general software design and development and these have become extremely popular.

Even though it is near 30 years, but still the applicability of this is huge in your own solutions that you work out, you will use them, you will see the existing solutions have made use of them and not only that, some of the solutions that we are going to discuss have become part of the programming languages. For example, while using JAVA, you might understand you might have questions in your mind that’s why it is done that way. For example, why is that in a collection, JAVA collection, the iterators are used, you have to create iterator object and then you use the iterator object to iterate among the individual elements of the collection classes. But for array we do not, we just traverse the array through without iterator, we use i = 0; i < n; i++. But then for collection classes we use iterator objects, we will discuss about the iterator pattern and we will see that this is straightaway used in the JAVA programming language and many of the GoF patterns are actually implemented in the JAVA language and many other programming languages. For example, the decorator pattern, these have been used in the JAVA IO.

You might wonder that why is JAVA IO is as it. But if you know the decorator pattern, you will have the answer to the question, that the JAVA IO is as it is, because it makes use very heavily the decorator pattern. Similarly, the composite pattern and so on. So, not only you can use the design patterns to arrive at your solution, design solution, it will help you understand somebody else design. But also, you will appreciate why JAVA programming language and other programming languages do certain things in some way.

(Refer Slide Time: 15:41)



"Gang of four" (GoF) and GRASP Patterns

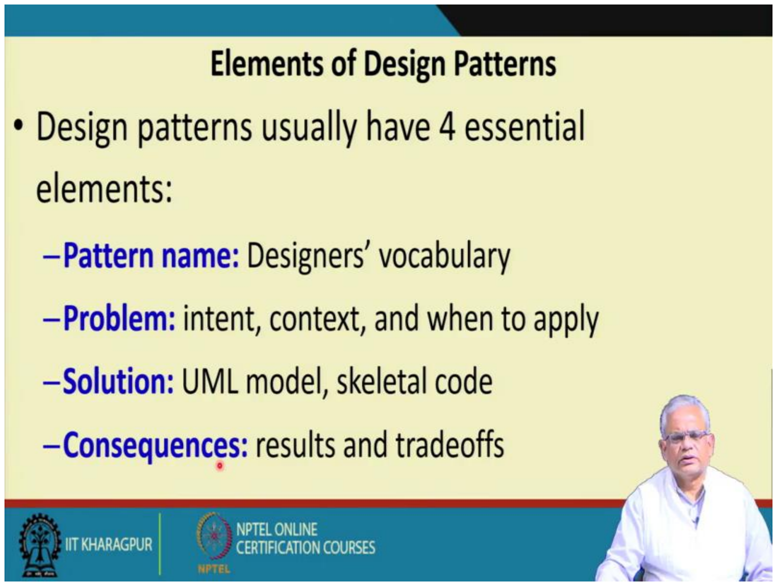
- Erich Gamma, Richard Helm, Ralph Johnson & John Vlissides (Addison-Wesley, 1995)
  - Design Patterns book catalogs 23 different patterns
- Larman
  - GRASP (General Responsibility Assignment Software Patterns) pattern
- Several other types of patterns are also popular.

Design Patterns
Elements of Reusable Object-Oriented Software
APPLYING UML AND PATTERNS
CRAIG LARMAN
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

For the GoF patterns that we discuss, we will largely fall on the book on Gamma, Helm, Johnson and Vlissides, this is the book which catalogs 23 different patterns. The number of hours that we have now may not be sufficient to cover all 23 different patterns. But we will definitely be able to cover 10 to 15 patterns. We will focus on the important patterns 10 to 15 important patterns and we will also discuss the GRASP pattern, all the GRASP patterns which are the, GRASP is the acronym for General Responsibility Assignment Software Patterns.

The GRASP patterns have been described in the book, applying UML and patterns by Craig Larman. But, as we said, there are several other patterns which are also popular, we will not have the opportunity to discuss them in this course, but we will encourage you to read them and get familiar with them, because if you are working in some domain definitely if you know the patterns, you will come up with better designs, you will be able to appreciate what is happening in some program much better if you know about the patterns.

(Refer Slide Time: 17:36)



Elements of Design Patterns

- Design patterns usually have 4 essential elements:
  - **Pattern name:** Designers' vocabulary
  - **Problem:** intent, context, and when to apply
  - **Solution:** UML model, skeletal code
  - **Consequences:** results and tradeoffs

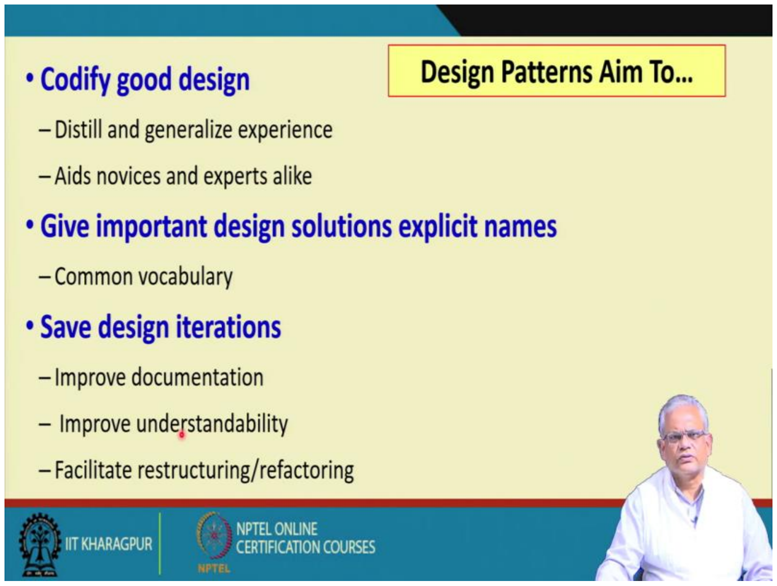
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL

Now, how do patterns appear? Patterns typically have four elements depending on the specific pattern we discuss; we try to give four elements to describe a pattern. One is of course, the pattern name, every pattern has a name and this forms a vocabulary for the designer, if you know some patterns like decorator, builder, Singleton, iterator etc., you can communicate with other designers or somebody may look at your design and say that see, why do not you use a decorator pattern or why do not you use a composite pattern here for this part of the design.

This helps to communicate ideas between different designers. Every pattern has a name and then the problem it is trying to solve, that is the intent of the pattern, the context in which the problem occurs and also some guidelines as to when to apply the pattern and the solution to the pattern problem is given typically in the form of a UML model, mostly a class diagram and also skeletal code and in our discussions we will discuss about JAVA code, but the solution can be given in other programming languages as well.

But since we have to discuss using any one language, we will use JAVA language for discussing the pattern solutions and UML model, which is the main part of the solution is typically given in the form of a class diagram and also we will have the final description of the pattern that is consequences of using this pattern. What are the tradeoffs made? What are the situations when this pattern solution should not be used and so on?

(Refer Slide Time: 20:09)



Design Patterns Aim To...

- **Codify good design**
  - Distill and generalize experience
  - Aids novices and experts alike
- **Give important design solutions explicit names**
  - Common vocabulary
- **Save design iterations**
  - Improve documentation
  - Improve understandability
  - Facilitate restructuring/refactoring

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL

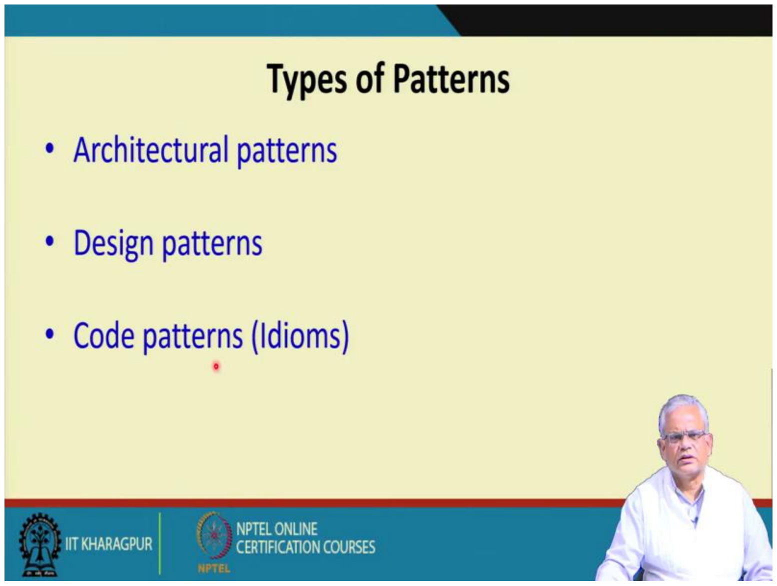
The design patterns, they aim to codify good design. The good designers, they have used these proposed and use it, we will see the wisdom they have in proposing the pattern solutions. Some of the pattern solutions are more intuitive. Specifically, the GRASP patterns are more intuitive. If we are experts in design, we will come across or will think of those patterns ourselves. But then there are other patterns, where the pattern solutions are a bit surprising and elegant.

Given the problem, to a good designer, it will be hard for him to come up with those solutions, which we will discuss, he might need hundreds of iterations, he will propose initially some solution, refine it and so on and after hundreds of iterations, maybe he will come up something like that. So, it will be a good idea to know those patterns, because you will save all those design iterations even if you are a good designer.

The patterns distill and generalize the solution which very experienced developers have proposed. It is useful for novices and experts alike as long as we understand the pattern, it will help us to improve our design. Another very important thing is that we will be familiar with the vocabulary and if you are working in a team environment or trying to communicate design ideas to somebody, these names become very important, you might go to a company and there somebody might say that, why do not you use the composite pattern in your design and that would make sense if you know this patterns.

This form a common vocabulary among the designers and developers. It saves design iterations as we are saying that sometimes you might come somewhere close to the solution given to the design patterns, but then you need hundreds of design iterations and months or years of time to arrive at such solution and therefore it saves time, iterations you come up straight with a good solution. It helps improve the documentation, improves understandability of the design and also the most important is that if you use the design patterns, your program becomes easily maintainable. You can restructure, refactor and so on.

(Refer Slide Time: 23:25)



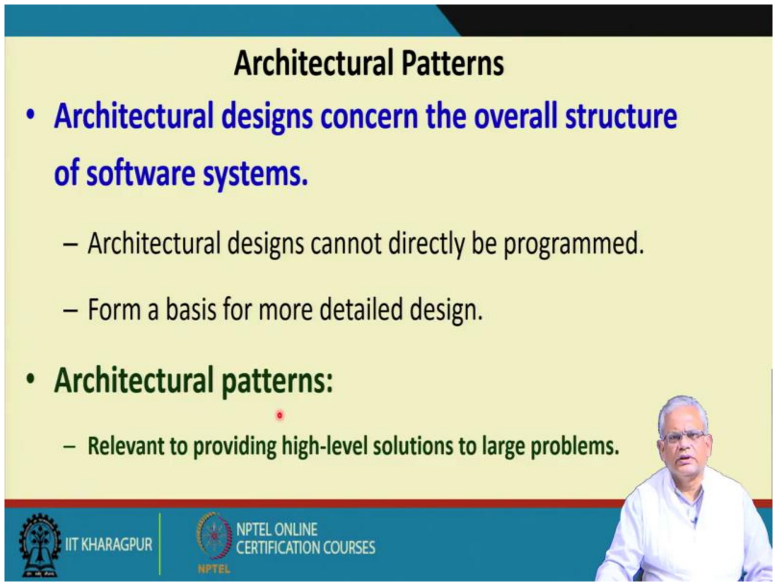
Types of Patterns

- Architectural patterns
- Design patterns
- Code patterns (Idioms)

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL

Now, what are the different patterns that exist in software development area? Broadly speaking, there are three main types of patterns, the architectural patterns, the design patterns or the code patterns, which are also called as idioms. In our discussions, we will focus on the design patterns. But since we are discussing the patterns, we will also have an idea of what are architectural patterns and code patterns.

(Refer Slide Time: 24:11)



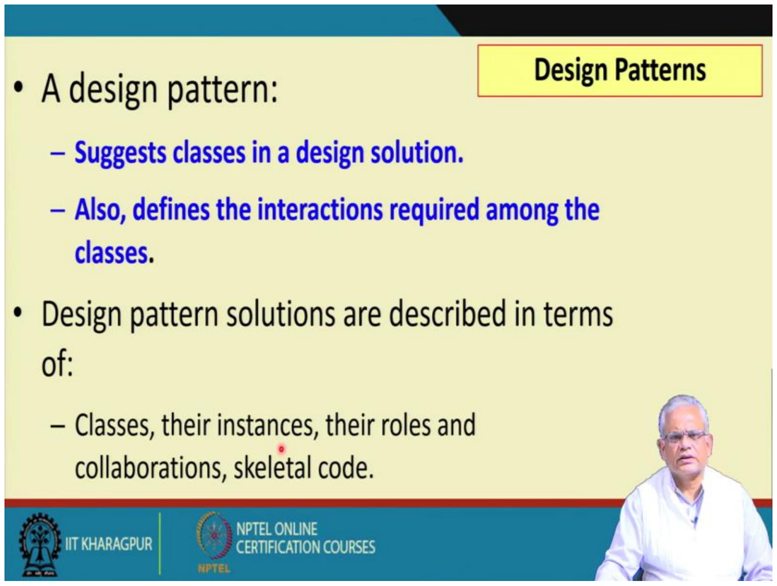
Architectural Patterns

- Architectural designs concern the overall structure of software systems.
  - Architectural designs cannot directly be programmed.
  - Form a basis for more detailed design.
- Architectural patterns:
  - Relevant to providing high-level solutions to large problems.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

The architectural patterns, these deal with the overall structure of the design, high level solutions and the high-level solutions do not directly translate to code. The high-level solutions guide the detailed design. The high-level solutions are the architectural patterns for high level solutions and we will discuss the MVC pattern which is an architectural pattern.

(Refer Slide Time: 25:00)



Design Patterns

- A design pattern:
  - Suggests classes in a design solution.
  - Also, defines the interactions required among the classes.
- Design pattern solutions are described in terms of:
  - Classes, their instances, their roles and collaborations, skeletal code.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

MVC model view controller is a very popular MVC pattern, we will just look at that. The design patterns are the ones which are provided a design solution and the design solution is broad, in that you can apply it to many problems, these are general and broad, broad applicability so that for a specific problem, you can just tailor it and apply it and after our discussions and after your initial use of this, I am sure that you will get convinced that the design patterns help improve the design.

Not only that, it provides a class diagram, but also it defines the interactions required among the classes. The pattern solutions for design pattern are given in form of class diagrams, objects, the roles of the classes, collaborations among classes, and also the skeletal JAVA code for every GoF pattern, so that once you have the design, you can easily write the code and also if you look at the code of a solution, you will be able to understand that what patterns the designer has used. We are almost at the end of this session. We will stop here and continue in the next session. Thank you.