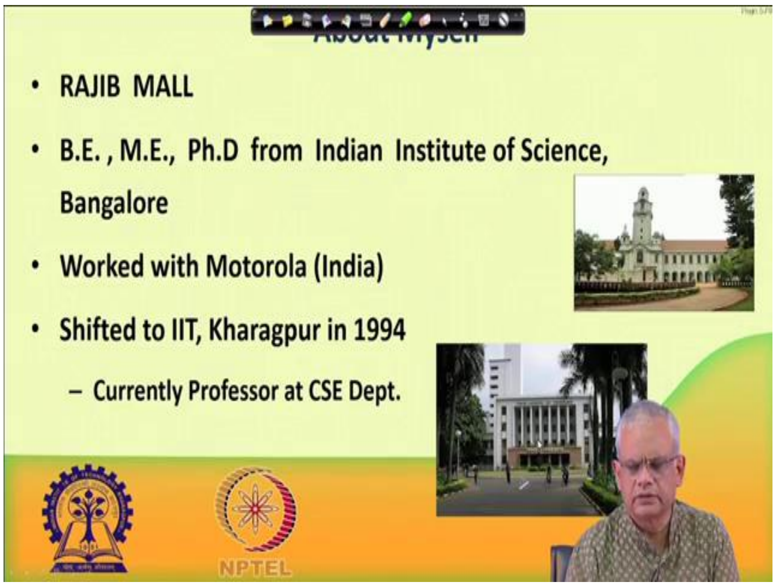
Software Project Management

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

Lecture - 01
Introduction - I

Welcome to the Software Project Management course. As has been displayed earlier the course will be handled by myself and Professor Durga Prasad Mohapatra. My name is Rajib Mall as you can see on the display. I belong to the Computer Science and Engineering Department of IIT Kharagpur. Today, we will have some introductory discussion.

(Refer Slide Time: 00:47)

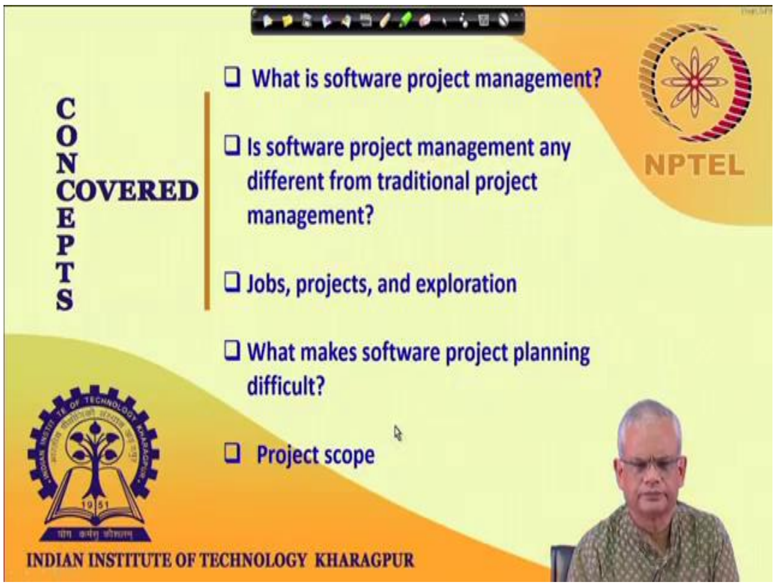


About Myself

- RAJIB MALL
- B.E., M.E., Ph.D from Indian Institute of Science, Bangalore
- Worked with Motorola (India)
- Shifted to IIT, Kharagpur in 1994
  - Currently Professor at CSE Dept.

First is about myself name is Rajib Mall, did all my education Bachelors, Masters, and PhD, from the Indian Institute of Science Bangalore. So, that is the institute studied and then worked for Motorola India for about 3 years. And, then shifted to IIT Kharagpur in 1994 and currently a professor at the CSE department, this is the Indian Institute of Technology Kharagpur building.

(Refer Slide Time: 01:31)


CONCEPTS COVERED

- What is software project management?
- Is software project management any different from traditional project management?
- Jobs, projects, and exploration
- What makes software project planning difficult?
- Project scope

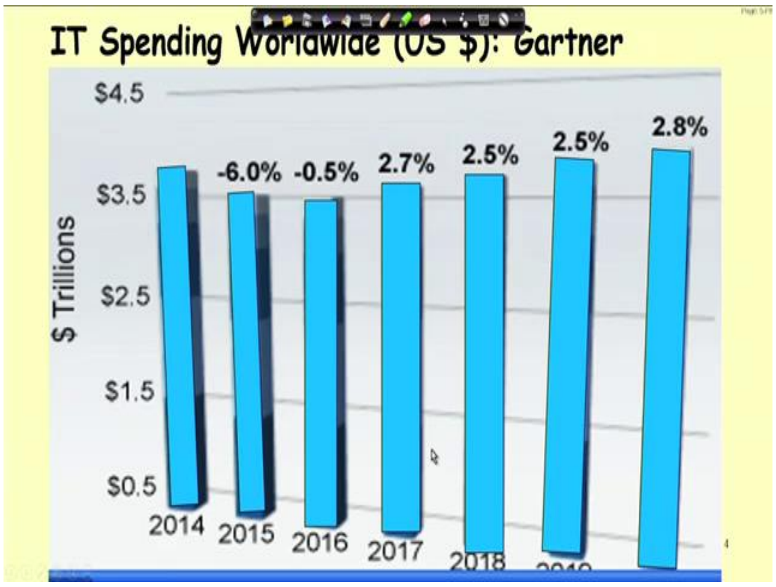
INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

NPTEL

The let us look at the topics that we will discuss in this introductory lecture. We will first address what is software project management? Is software project management any different from the traditional project management? The difference between jobs, projects, and exploration work.

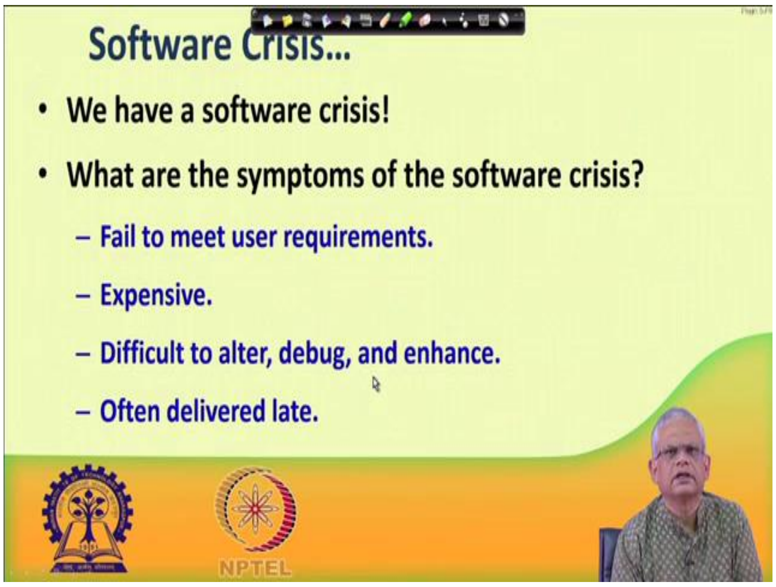
What makes software project planning difficult? And, what do we mean by project scope? Why is it important and how does one define the project scope? So, these are the basic topics we will discuss in this introductory lecture.

(Refer Slide Time: 02:24)


IT Spending Worldwide (US $): Gartner

The IT spending worldwide is huge based on a Gartner report roughly about 3.5 trillion dollars in 2014. And, gradually increase to about 4 trillion dollar and compare this with the gross domestic product of India is about 2.5 trillion dollars. So, the IT spending is even much more than all the domestic production of India is a huge.

(Refer Slide Time: 03:20)



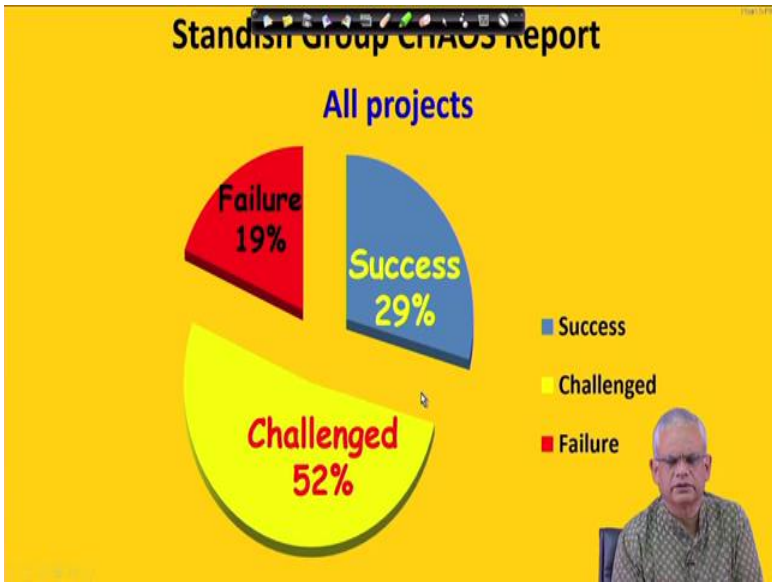
Software Crisis...

- We have a software crisis!
- What are the symptoms of the software crisis?
  - Fail to meet user requirements.
  - Expensive.
  - Difficult to alter, debug, and enhance.
  - Often delivered late.

And therefore, software has become a very prominent part of our living, we spend huge amounts of money software; we encounter software in many places. And, especially in India we are known as a software powerhouse, huge amount of software gets developed, large number of companies and they undertake large number of projects to develop software. So, it is a important topic all over world. And, especially for India where large part of the software development takes place, but then everybody says that there is a software crisis. But what is this crisis? Let us first get to understand the crisis.

What are the symptoms of the crisis? If we look at the symptoms of the crisis that the software that had developed and delivered to the customer very often fail to meet the requirements of the customer, extremely expensive these are difficult to alter, debug and enhance and also these are delivered late to the customer. This is unlike hardware where you get the hardware quickly developed and given. So, let us see, what is this after crisis, what causes it, and how to overcome?

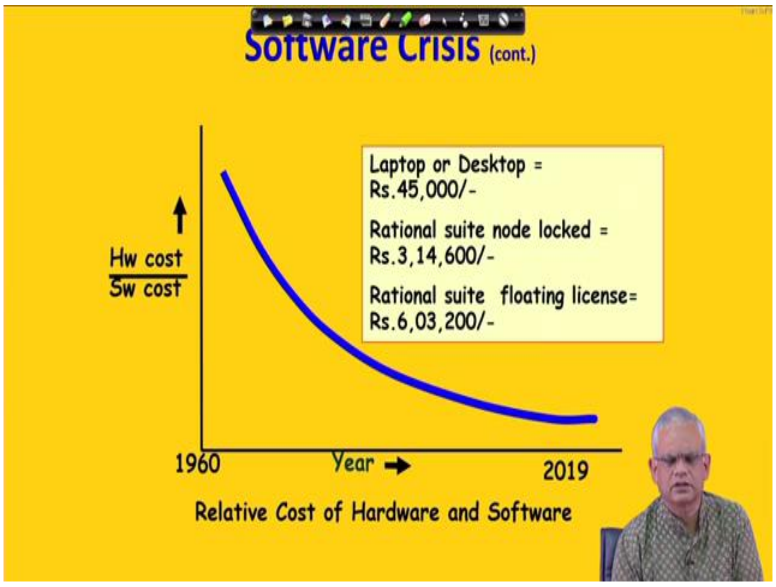
(Refer Slide Time: 05:16)


Standish Group CHAOS Report

All projects

If we look at the Standish group report of all projects roughly about a third we can say success of all the projects taken all over the world about 30 percent successful projects. And, about 20 percent are outright failure nothing comes out of that project. And, roughly about half are challenged projects, which are delayed cost escalation, poor quality and so on. Let us see what is the reason behind such a dismal figure, that only one third of the projects is successful and 20 percent are failure and so on.

(Refer Slide Time: 06:17)


Software Crisis (cont.)

Laptop or Desktop = Rs.45,000/-

Rational suite node locked = Rs.3,14,600/-

Rational suite floating license= Rs.6,03,200/-

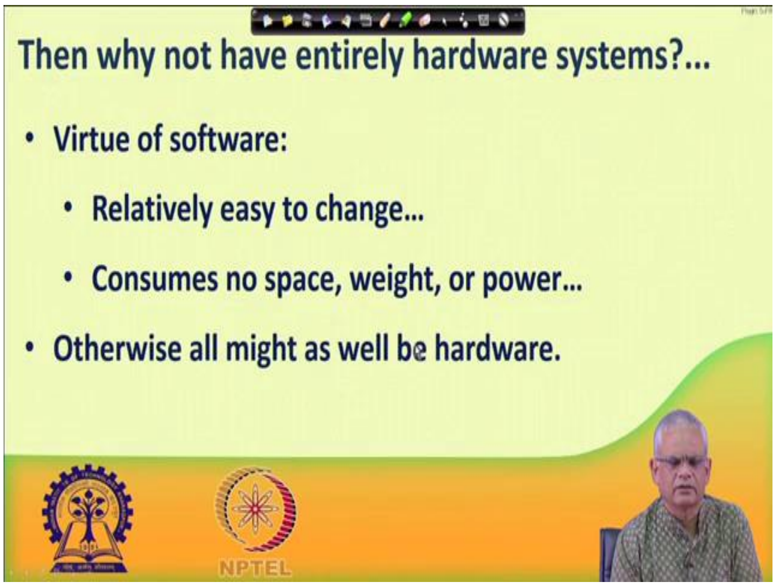
Relative Cost of Hardware and Software

If we look at the scenario now, we can see that the amount that the companies spend on hardware versus the amount of they spend on software, the ratio is drastically reducing. What it means is that companies are spending less on hardware and more on software or in other words software costs are increasing very rapidly compared to hardware costs. The companies spend a large part of their budget on IT on software, just to give an example, that how the software cost is much more than the hardware cost look at a desktop or a laptop, if you buy it from the market.

The hardware and also with its operating system and so on is 45,000. So, basically the raw hardware will be much less than that. But, then you get the laptop or desktop and let us say you want to do some software development work on that and you buy a case tool. If, you buy a rational suite node locked costs 3,00,000, just look at this that the raw hardware for the laptop or desktop is much less than 45,000, because 45,000 also includes the software operating system and so on.

And on that you want to run software development tool costing about 3,00,000, that is a node locked a floating license will be much more. So, just look at the irony that you buy the hardware in such a low price and the software that you run on it just one example there are many software, that you might have to buy. So, looks like that, the software is becoming, so, expensive that the hardware cost will become almost negligible. And, you buy the software and the hardware comes free with it soon that situation is going to come, that you buy the software and that comes preloaded on a hardware. The hardware is free basically, but then what is the problem that is causing this.

(Refer Slide Time: 09:12)



Then why not have entirely hardware systems?...

- Virtue of software:
  - Relatively easy to change...
  - Consumes no space, weight, or power...
- Otherwise all might as well be hardware.

But, before that let's see that if the hardware is such a nice thing that costs very less delivered promptly and we said that the cost is almost negligible to software. Then, why not have only hardware systems entirely on hardware system. Because, anything that can run on software, anything that we can do with software, we can also do with hardware, we can design the hardware to do whatever we were doing software. For example, let us say a word processing software. We can even have a small hardware component entirely hardware, which can also do this word processing.

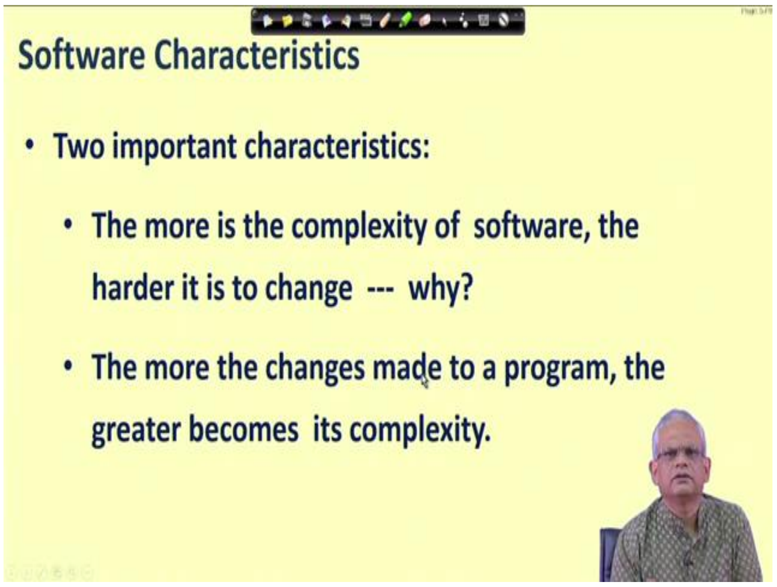
Let us answer this very basic question, that why not have an entirely hardware system, get rid of software it is problematic, expensive lot of bugs, delivered late and so on. But, then there are some virtues of software and because of that everybody is using software, otherwise there will be entirely hardware systems. Let us look at the virtues of software because of which software is preferred by customers, rather than having an entirely hardware system. The first virtue is relatively easy to change, you want to add additional functionality, modify a functionality, the next version can appear quickly you give your request might send you a patch and so on.

So, the software is very easy to change just do a code change, recompile and that is done. The other big advantage is that consumes no space, it has no weight or intrinsically there is no power associated with software. Just imagine that, if you are running a word processing software sorry you were doing some word processing. And, then you developed hardware for word processing, then you wanted a excel and you had to get another hardware for it.

And, you want to do a image manipulation software sorry image manipulation functionality and you had another piece of hardware and slowly your hardware will occupy the entire room not only the space, but the weight and imagine all of these will be consuming power and that is a big disadvantage for hardware.

But, software even for very complex functionality where the hardware will be extremely large, it occupies no space, weight, or intrinsic power that runs on the hardware. And, that is the reason why the software is becoming more and more important in our life, more and more functionalities are getting implemented on hardware on software. So, there is a basic hardware on which all our new functionalities that are required are developed by software.

(Refer Slide Time: 13:01)



Software Characteristics

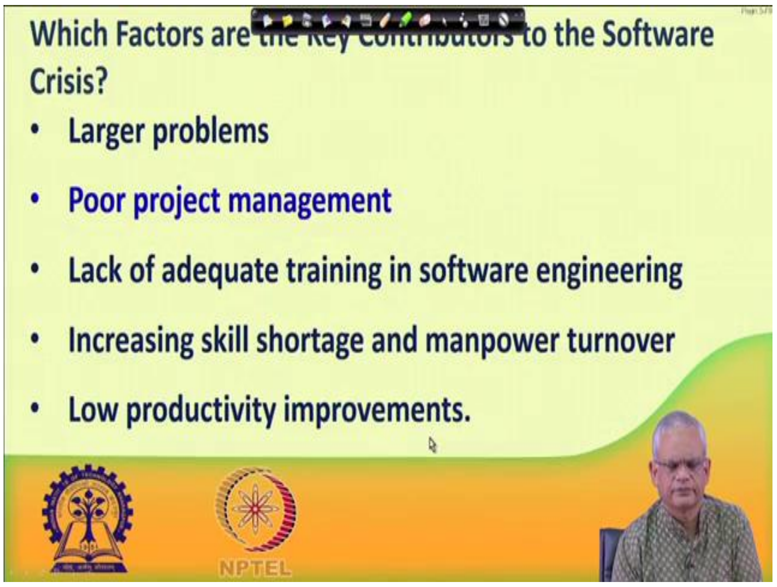
- Two important characteristics:
  - The more is the complexity of software, the harder it is to change --- why?
  - The more the changes made to a program, the greater becomes its complexity.

Now, let us look at some characteristics of software that are very important to our subsequent discussions. There are many characteristics, but we have singled out only two characteristics; one is that, the more complex is a software, the more hard the more harder it is to change, why is that? The reason is not very far to say. To be able to change the software we must first understand the software, we must first understand where the change has to be done, which variables are to be changed, what code is to be written for that and so on.

The more complex is a software, the more time effort is necessary to understand the software and find out where exactly and what change needs to occur. The second thing is that each time we make a change to a software, the greater becomes it is complexity. Each time we change a software the change may be required due to various reasons, maybe to fix a bug, maybe to enhance the functionality, maybe to make it a better performance and so on. There are 100s of reasons why a specific software would need to change, and for each change, the software becomes more complex.

Why is that, the reason is that the small changes that occur. For example, bug fix for example, adding small functionalities and so on, we just make ad hoc changes to the software. The initial software is developed with a good design and so on, but each small change deteriorates or makes the design worse, these are developed as small fixes. And, therefore, these make the software more complex, more difficult to understand, more difficult to change and so on.

(Refer Slide Time: 16:12)



Which Factors are the Key Contributors to the Software Crisis?

- Larger problems
- Poor project management
- Lack of adequate training in software engineering
- Increasing skill shortage and manpower turnover
- Low productivity improvements.

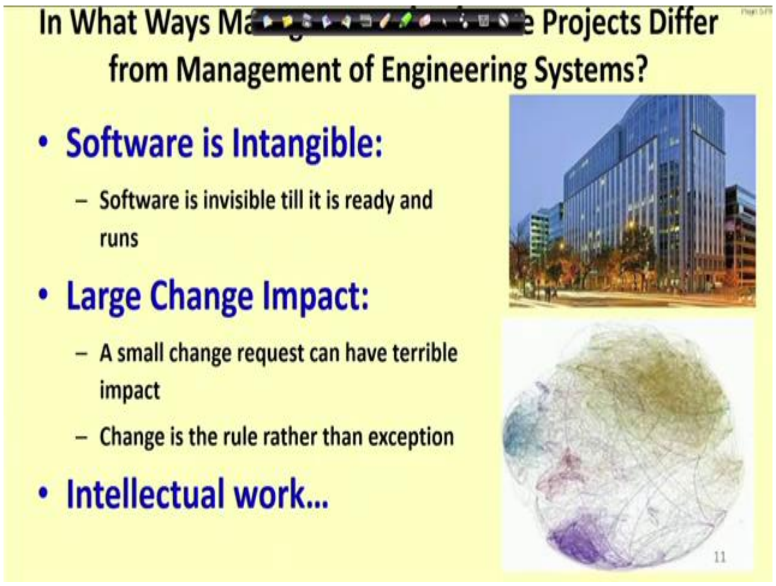
Now, let us look at the basic question that we have been trying to address in this lecture is that, what is the reason behind software crisis? We said the software crisis as symptoms, which show up as poor quality software, delayed projects, project failure and so on costly and so on. There are many reasons which contribute to the software crisis; the first one is larger problems. Now, the software has large number of functionalities

100s or 1000s of functionalities or 10s of 1000s of functionalities for even simple software packages.

Poor project management; project management is an area of concern because many projects they get developed without a proper project manager and without proper project management skills. And, this happens to be one of the major reason for the software crisis, why the projects fail delivered late and so on. The third point is lack of adequate training in software engineering. The developers start developing with some domain knowledge, but then they ignore the software engineering issues design requirements specification, testing, change management and so on.

Increasing skill shortage manpower turnover, halfway the project, the project is halfway and many of the key persons leave quite common and that delays the project. And, also other reason is low productivity improvements. Largely software has remained manual work even with all the automation automated tools and so on. Still lot of code needs to be written manually, lot of testing has to be done manually and so on.

(Refer Slide Time: 18:59)



In What Ways Ma[illegible]e Projects Differ from Management of Engineering Systems?

- **Software is Intangible:**
  - Software is invisible till it is ready and runs
- **Large Change Impact:**
  - A small change request can have terrible impact
  - Change is the rule rather than exception
- **Intellectual work...**

11

But, then let us understand another very basic question, that in what ways software project management differs from management of other engineering projects. Let me just repeat that, what is the main difference between software project management and management of other projects non software projects? The first problem is that software is intangible. You do not see the software until you see that it is complete and running then only you see that screens are getting displayed it does something, but until that happens, it is just a set of documents or some pieces of code, you cannot make out that how much is remaining just by looking at the documents or the code. Maybe huge amount of code has been taken has been written, but then the software is far from complete, because many bugs needs to be fixed it needs to be tested and so on.

What is difficult, what is we are unable to see is also difficult to manage? For example, construction of a large building, here the project manager can estimate how much effort time it is going to take to construct the building maybe 6 months. Huge building 6 months and then he can see that the building is coming up the external part is complete, the internal needs to be done and so on. At any time he can accurately tell that how much work is remaining, but in software that is not the case.

The second problem is change impact. In a building you make small alterations, let us say building project. You know that, what exactly will be the impact of that and you can estimate that and make the small change that may be required. But on the other hand and the software is extremely complex, you change something, you try to change something and you see that many other things are not working or working in a wrong way. So, the impact of a change is very easy to identify in case of other projects, but in a software project it is so, complex that you cannot estimate what will be the impact of changes?

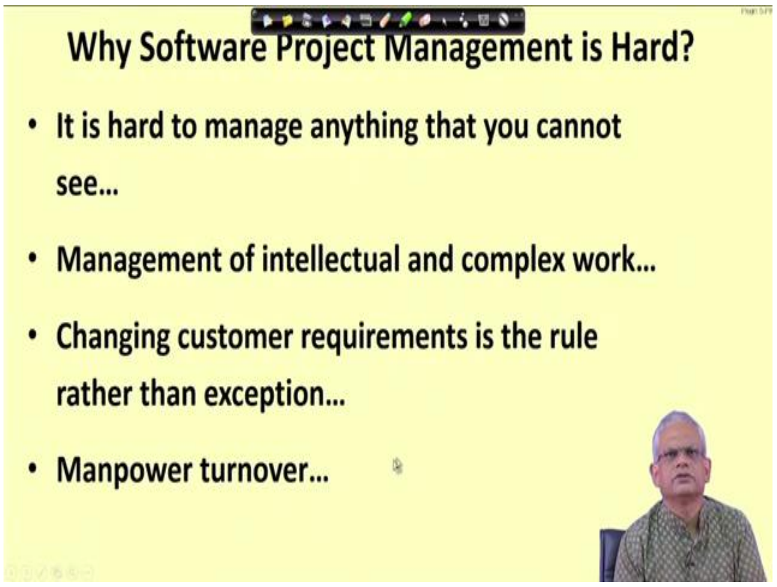
But, unfortunately the software is required to change rapidly. And, that is one of the advantages of software we said that you can change it quickly. And, therefore, the change requests are millions time more in software as compared to hardware, when you are developing a hardware you think twice before giving a change request. Because, that will be very hard to incorporate in hardware, it has to be totally redone. Whereas, everybody knows that software can be changed and therefore, lot of change requests come.

The third problem is that software projects are intellectual work. Compared to a building construction, where the bricks have to be laid the roof has to be constructed, boundary wall has to be constructed, painted and so on. All these are largely manual work and manual work is easy to estimate, you can easily find out how much a painter can paint per day or a labourer can lay the bricks per day and so on.

Whereas, software is a intellectual work. It is very difficult to estimate what is the complexity of the work that somebody is doing and how much he will be able to accomplish. So, these are the three main differences between other projects and software projects. The first thing is you have to manage something, which is invisible. And, anything that is invisible is difficult to manage. The, second thing is that changes are very frequent in software projects, but then it is also very difficult to estimate the change impact, the change impact is usually very large and it is also difficult to estimate.

The third thing is that, we have to in software project management; we have to manage intellectual work. This is a much bigger problem than managing manual work. In manual work we can easily estimate, how much can get down on a day, what is the complexity of the problem and so on. In intellectual work we can neither estimate what is the complexity of the work and neither how much effort will be required for that.

(Refer Slide Time: 25:45)



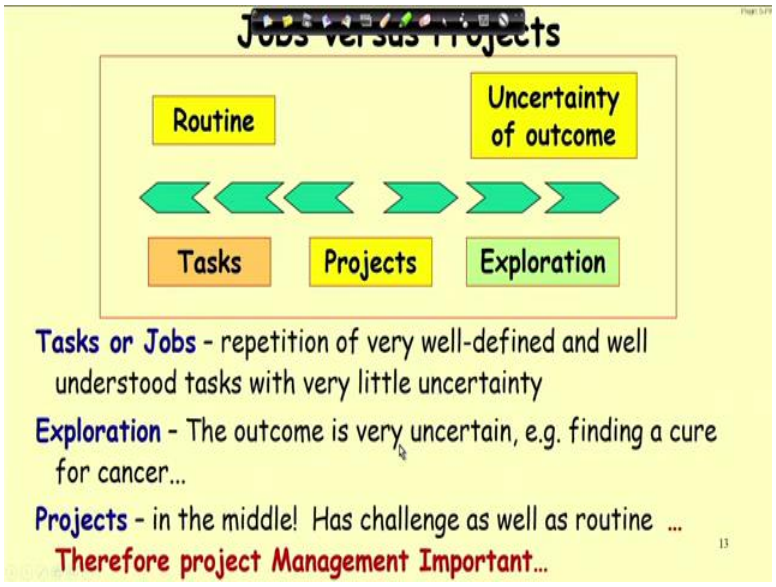
Why Software Project Management is Hard?

- It is hard to manage anything that you cannot see...
- Management of intellectual and complex work...
- Changing customer requirements is the rule rather than exception...
- Manpower turnover...

And, that brings us to the question, that the software project management is hard as we might have guessed from the previous discussion. It is much harder than managing other types of projects. Software is invisible and it is hard to manage anything that you cannot see, it is management of intellectual and complex work, changing customer requirements is the rule rather than exception, and also the manpower turnover. You may be a project manager on some project, but then you find that some of your key developers, they leave the project.

And, they have the domain knowledge they have developed and once they leave it becomes very difficult to get another person who will understand his work and start developing from that point. These are a few of the major problem a software project manager has to face. But, then there are many other problems here smaller problems which we have not listed.

(Refer Slide Time: 27:10)



Jobs versus Projects

Routine
Uncertainty of outcome
Tasks
Projects
Exploration

Tasks or 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 ...

Therefore project Management Important...

13

But, before that we like to discuss before proceeding further, we like to discuss, what are projects? How is it different from tasks and exploration? The tasks are routine jobs, for example, let us say buy a movie ticket or watch a video lecture. These are tasks or jobs, these are repetition of very well defined well understood activity with very little uncertainty, watching a lecture yes you know that you have to sit there and watch can be done you have to sit there half an hour whereas, exploration is on the other end where the outcome is very uncertain.

For example, finding a cure for cancer or finding a solution to the global warming. So, these are exploration the outcome is uncertain you never know, whether your work will have a success or not. The projects are in between the projects has challenge as well as there are some routine tasks involved. And, that is the reason why project management is important here, because for the tasks or the exploration you do not need project management. These are purely innovative work whereas, projects are the one where you need project management that is important thing here, with this introductory discussion, we will stop here and continue from this point in the next lecture.

Thank you.