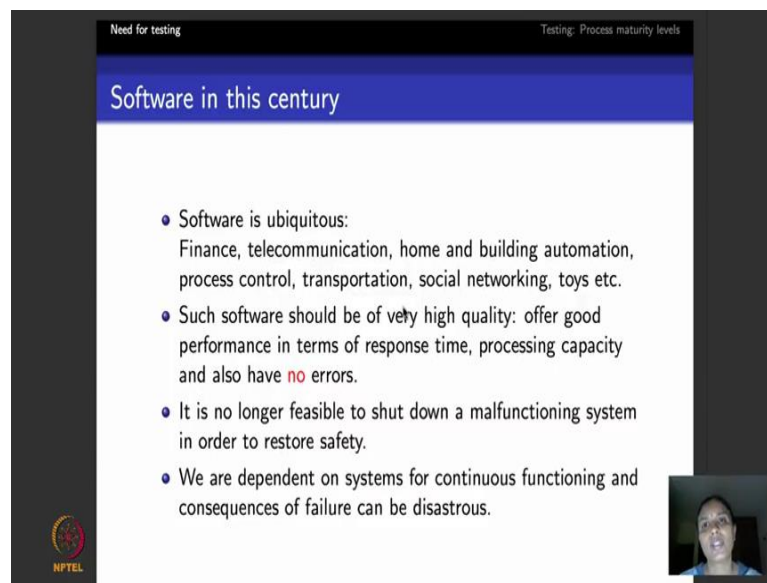**Software Testing**
Prof. Meenakshi D'Souza
Department of Computer Science and Engineering
International Institute of Information Technology, Bangalore

Lecture – 01
Software Testing: Motivation

Hello everyone my name is Meenakshi. And this is a first lecture that I will be doing as a part of the course on software testing that I am currently offering for NPTEL. So, what are we going to do today? Basically I will talk to you about the kinds of software that we encounter in the present world, and what do expensive errors look like, and how to avoid these expensive errors by using testing. So, it is basically motivation module where we deal with how important testing is.

(Refer Slide Time: 00:45)

So, what a software in the current century look like? If you look all around you, it is not too difficult to understand that there is software.

So, we use Paytm, we use banks online, we go to an ATM machine there is software, I make a call to my friend or my parents using my mobile phones, there is software. Software that controls the heating, ventilation then air conditioning in our homes and offices, there is software that tells you how electricity flows through a grids, how water flows through a network. Software that run in manufacturing industries, the software that help us to drive a car, autopilot and other kinds of software that help you to fly our

planes, when the software that controls real network, the software even in toys that children use right.

So, what is our expectation about such software? We all expect the software to be error free, not only that we want the software to respond fast right. The second I insert my card into the ATM machine and what the welcome screen the next second before I had link my id, I cannot be it long enough. And the other important thing to notice that if you consider a software like the autopilot of a plane, let us say there is an error in the software you cannot say that I will shut it down mid flight, I will rectify the bug and I will restart to be able to run right. It is no longer feasible to be able to shut down system because there is an error in the software. We want the system to be able to continue to done in the autopilot case it needs to be able to continue to fly the aircraft safely and help us to land safely.
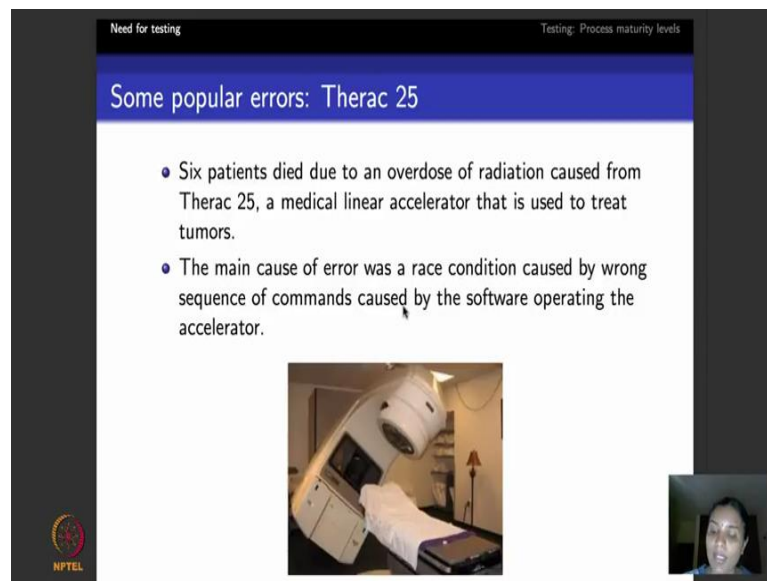
(Refer Slide Time: 02:22)

So, I would like to talk to you about some popular errors that have occurred in the past. This was the error is the error is the error in European space agency rocket called Ariane 5, it occurred exactly 21 years ago and the important thing to note that this error is because of a software bug. So, there was this rocket called Ariane 5 which was being controlled by software that is launched by European space agency.

So, this software had the following error. So, it was trying to squeeze in data corresponding to a 64-bit floating point number into a memory space that is allotted for a

16-bit integer. So obviously it is not going to be able to succeed in doing that. And because such rockets have safety critical systems they always have backup software, but in this case the problem was the backup software also had exactly the same error. So, this resulted in transmission of incorrect altitude data to the aircraft and this rocket Ariane 5, went and plunged into the Atlantic Ocean within 36 seconds after it was launched. So, that is about 15 years of total effort and you can imagine that millions of Euros that would have been lost.
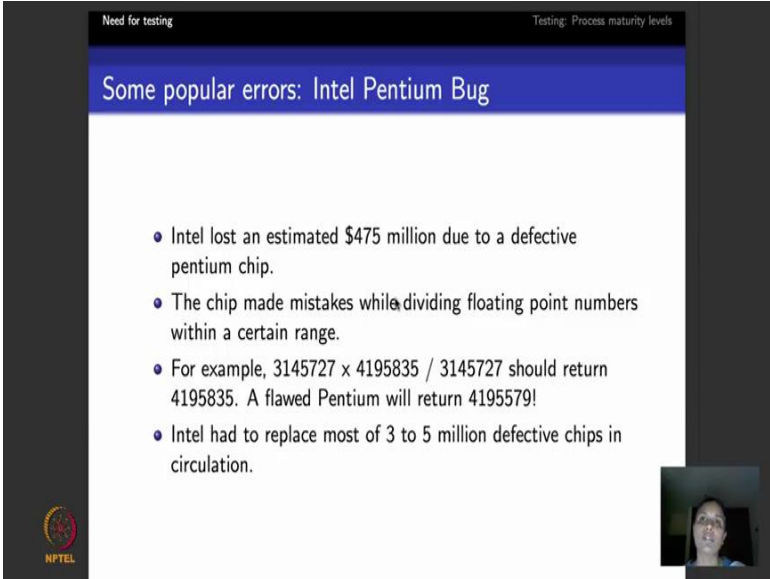
(Refer Slide Time: 03:33)

So, the next example that I would like to discuss about is another unfortunate example that happened, again because of a software error. So, in this case 6 patients lost their life due to buggy software in a machine that gives radiation therapy to cancer patients. So, there was a race condition error in this software. So, this software ended up calculating more dosage of radiation then what was needed and unfortunately these patients lost their life because of an overdose of radiation.

(Refer Slide Time: 04:05)
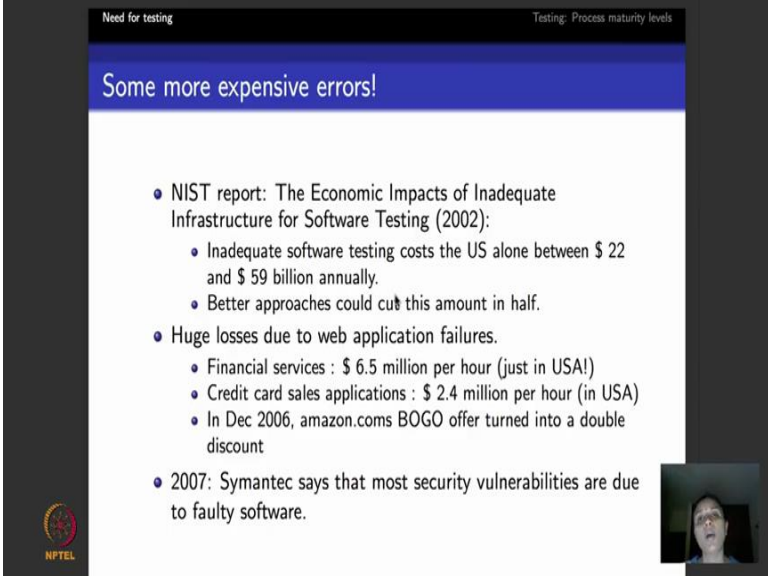
So, the next kind of example that I would like to discuss with you it is software that is expensive that costssomebody a lot. You all would have heard of this Intel Pentium processors right. So, there was this particular P4 Pentium processor, the mathematician in the US, when he was trying to do research towards its prime numbers, he found that this Pentium 4 processor was doing floating point division wrongly. So, when it was announced an Intel investigator did realize that all the P4 processors that it had released to the market had the same error. So, the only solution left was to be able to recall all these processors and that cost Intel a lot of money.

So, this does not bring us to the end of expensive errors. So, if you Google you know you can talk, you can find Toyota breaks, crashes all kinds of crashes happened.

(Refer Slide Time: 04:56)

So, here are some few reports that talk about how expensive errors can be and what is the cost of inadequate testing. National institute standards and technology in the year 2002 released a report which basically discussed the impact of software testing in US.

So, it says that inadequate software testing causes the US economy anywhere between 22 and 59 billion every year. Not only that the report goes ahead and says that if there are better approaches that can be found to do software testing, then you could bring down the amount of these losses to almost half of what it is. The other popular categories of losses are due to web applications which we use all the time. So, one particular thing that I would like to discuss with you now happened almost 10 years ago. Amazon had this online discount sale, and because of a software error what happened was that it ended up giving double the amount of discount and by the time Amazon realized it, it is too late that are already lost lot of money.

The next fact that I would like to drought your attention is a report that Symantec, the security organization published in 2007. So, it says that security related vulnerabilities that occur in a financial transactions, in our online wallets and so on mainly occur no because of cryptographic errors, they occur because of software errors.

(Refer Slide Time: 06:21)

So, we know expensive errors and software can be. So, let us try to look at the various kinds of software and typically how they are developed. You might have heard this buzzword called agile methodologies right. So, agile methodologies basically insist that people who develop the code, that is the developers, also have to unit test a code. Typically, developers do not have any great knowledge on testing, but agile methodology believes that the developer himself or herself the best person to identify the error in the code. So, that puts a lot of pressure on developers to be able to know testing well.

Now, let us look at the various kinds of software that we will deal with. If you consider a software that typically runs in your cars that helps you to do breaking automatically, or does a cruise control or a software that runs in your aeroplanes, such software is what is called embedded control software. So obviously, such software runs in what are called safety critical systems where a failure can be catastrophic. So, safety critical software have to meet regulatory standards where, by which regulatory authorities decide if a software has been tested enough and it is fit for release. So, this just means that almost double the effort goes into testing the software.

Other kind of software is what is called enterprise software. So, what would be a typical example of an enterprise software let us see a software that run city bank right or a software that managers are Indian railways ticketing system right. What do such software

have to deal with, the basically the first complex thing is that they deal with fairly huge data basis that have a lot of data and the software is critically dependent on the server and backup server running all the time right.

The third popular category of software is what is called web applications. Things like social networking sites, amazon online and so on and so forth. We all know how important it is for these software to be correct. Finally, I would like to end this slide with the point what free software suppose you pick up a piece of software from the internet for free. Just because it is available for free you are not willing to accept the fact this software could have errors. So, paradoxically we expect free software also to be error free.

(Refer Slide Time: 08:41)
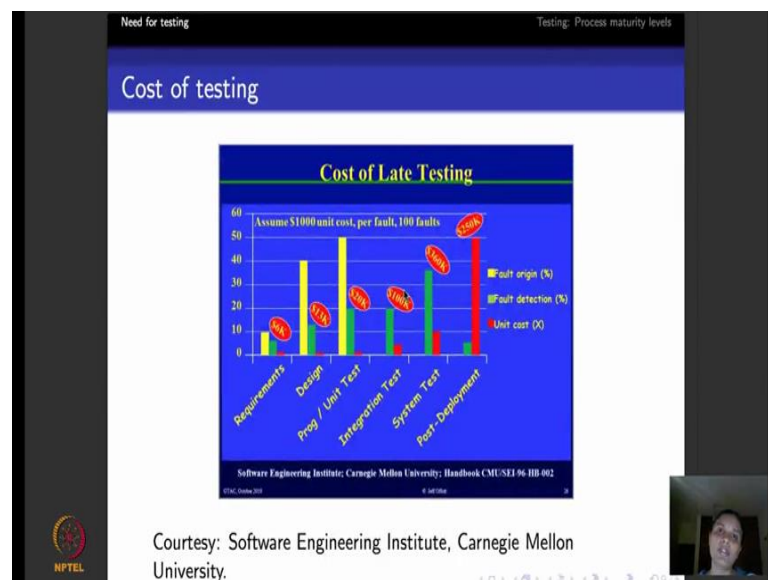
This is a very important slide. So, this is a part of study that was done at Carnegie Mellon university a few years ago. So, this slide discusses about how expensive testing can get as we go down software development.

So, if you see a typical software development initially you write requirements and then you do design right. And then your unit test your software and then put together the modules, that you have tested and do what is called integration testing then you put the software with the system or hardware that it is supposed to run on and in your system testing and finally, release the software.

So, suppose there was an error in the requirements, and it was found that and there then what the slide tells you is that gives you the cost - the cost is fairly low right, but suppose there was an error that was found in requirements or design, and it went all the way through testing, integration testing, system testing, did not get detected at all. The software called released and the error was found there. Then what this slide tells you is that the cost fixing that can be very high, it is not only the cost fixing the error the consequences of that error as we saw through the examples can also be really bad right.

(Refer Slide Time: 09:49)
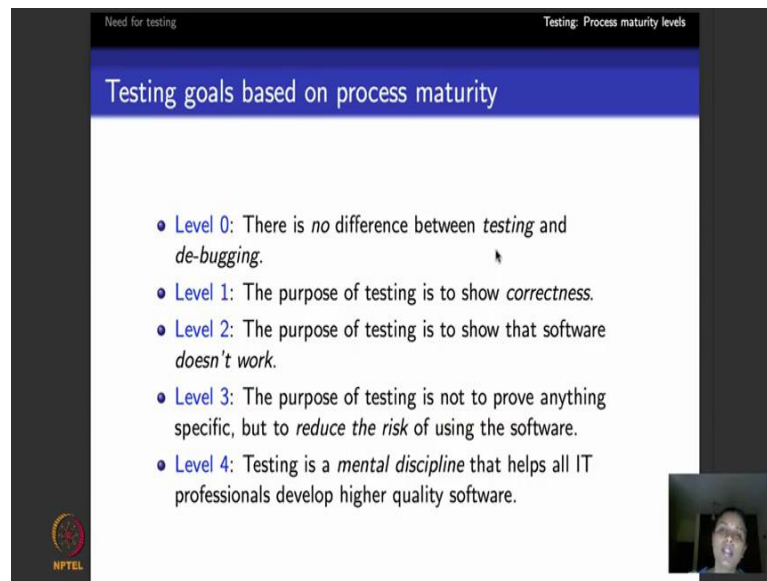

So, what are the facts and myths about testing? There is a popular saying by Edgar Dijkstra who said that never think that you can use testing and say that I have proved my software to be correct. It is wrong to say the testing prove software to be correct. The main goal of testing is to be able to find errors in the software right. Using testing I can never say that I have tested my software and it is fully correct, it is a wrong statement to make. The other wrong statement to make is that a developer typically thinks that you know " have written the code, I have debugged it well and my quality auditor my SQA in my company has inspected the code. So, there is no need to test it". That typically will not work; it still needs to be tested.

Another popular thought that people have especially these people who sell software testing tools is to be able to say that you just download and install my tool it can do magic; it will do every kind of testing you can think of. That is not true.

The main goal of testing tools is to be able to help you execute the test cases and record the result. Now to be able to design test cases, you will have to be able to design cases to find errors effectively. Typical Pareto principle applies here 80 percent of the errors come in 20 percent of the code or design. So, test case design which is what the course is about needs a human to be able to do it right.
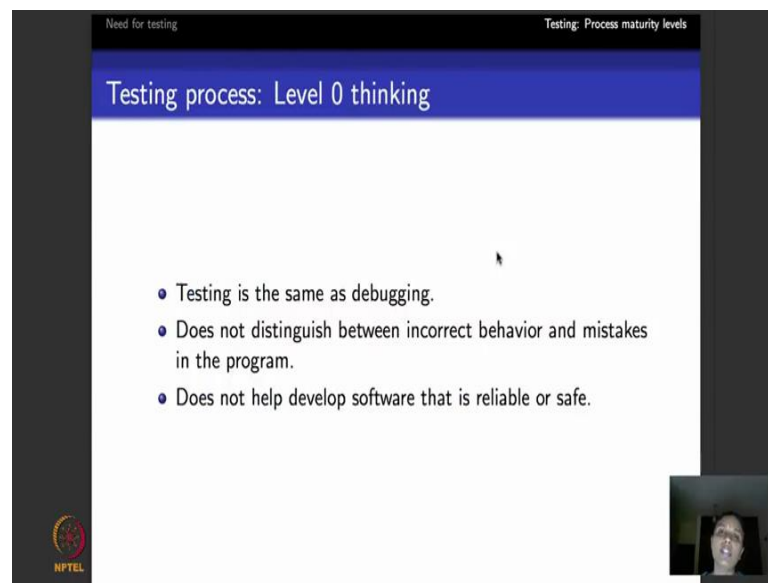
(Refer Slide Time: 11:18)

So, I would like to end this module by discussing about certain process maturity levels in testing. Why is it important to look at process maturity levels? It is important because it tells you what role testing plays in the software development that you do. Broadly there are 5 levels beginning from 0 and going all the way till 4.

(Refer Slide Time: 11:42)
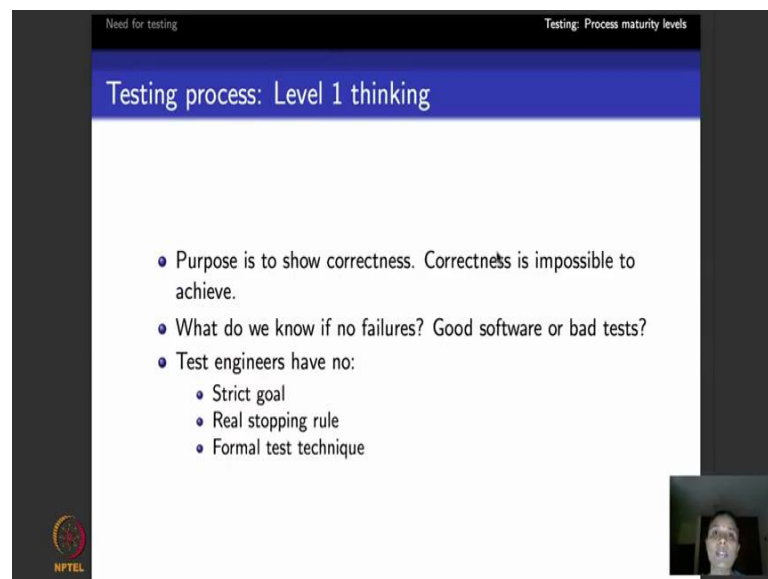
So, what does level 0 tell you? Level 0 tells you that there is basically nothing called testing. Developers write their code, they debug their code they ensure that it is correct and that is it, they go ahead and release it right. So, it is clear that this does not really help to develop software that is considered to be fully safe and reliable.

(Refer Slide Time: 12:01)
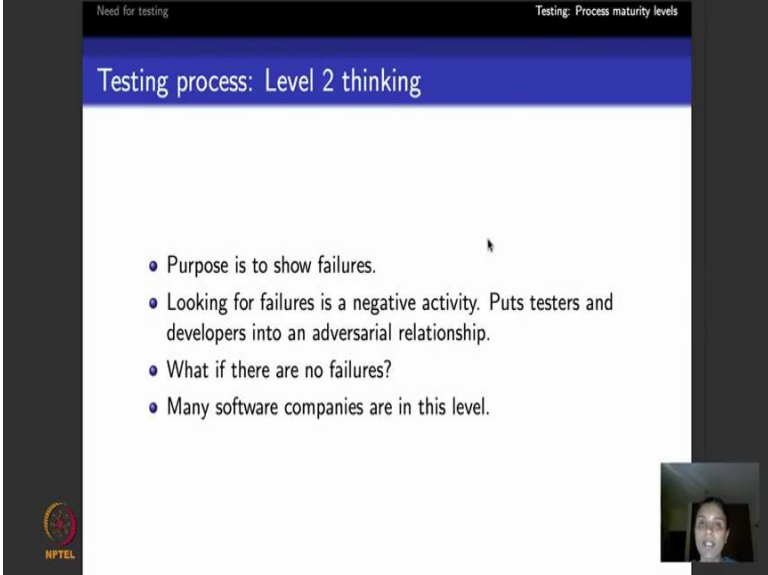

The next comes level one thinking. Level one thinking a developer thinks that he or she has written a piece of code and their goal to test is to be able to show that the code is correct. As we saw little while ago I clearly cannot use testing to show that a piece of

software it is correct right. So, let us see a particular piece of code has an integer variable. To be able to exhaustively tested on the 32 bit processor I need to be able to give every value which is 2 power minus 32 to 2 power 32 plus or minus 1, and even when an extra fast PC this is going to be able to take several years to do. So, unless I test it for every value I cannot say that the testing is correct.

So, here there is nothing like test engineers and they even if there are they do not have goals and they just show that the software is not failed, but the underlined listen it is not failed because it is correct or it is not failed because you have designed the test cases wrongly, that is never clear.
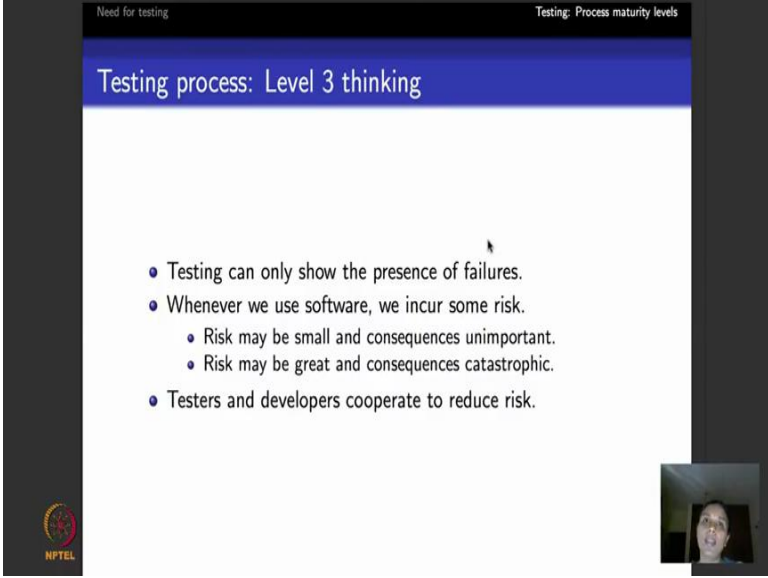
(Refer Slide Time: 13:00)

The next level is level 2 thinking when you begin to believe that the goal of testing is to be able to identify failures or errors in the software. This is the beginning of positively using testing, but it organizations that are typically at this level there is lot of tiff between developers and testers, they belong to different teams and then one does not want to help the other, and there is confusion even though the goal of testing is fully realized. We move on to level 3 where they not only realize that the goal of testing is to find errors, but they also work together and say that we will not only find errors, we will make sure that we reduce errors to the extent possible in the software.

(Refer Slide Time: 13:28)
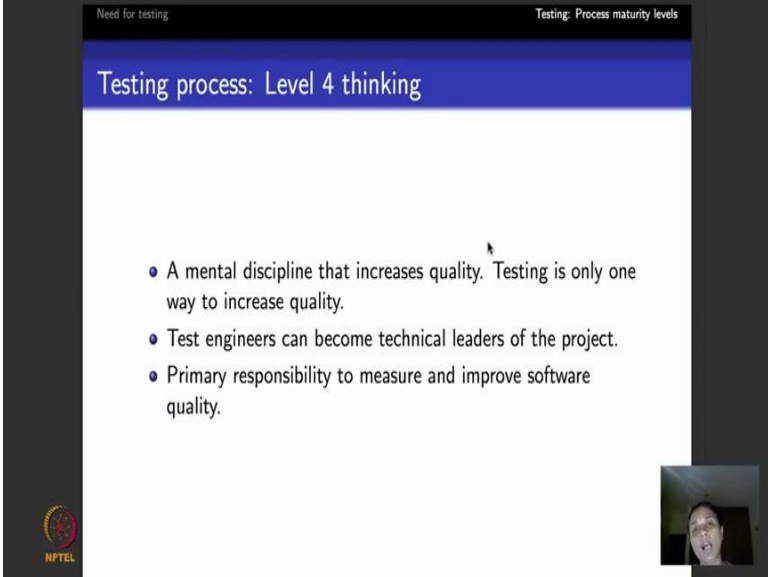
And that is also conscious understanding that when I release the software there is a bit of risk involved. The risk could be high or the risk could be low and my goal is to make the risk as low as possible.

(Refer Slide Time: 13:55)

Level 4 thinking is where large organizations strive to be in, here testing becomes a mental discipline. So, there is positive thought and efforts in the organization level to make sure that testing teams' efforts are taken into consideration to develop software that is as error free as possible. An organization like Microsoft which try to achieve

excellence in level 4 thinking and this is what is the desired level that we would like to achieve in testing.

(Refer Slide Time: 14:28)

Now, what is this course got to do with all the levels and terminologies that we saw till now? So, this course will help you to think that I want to be able to do testing at levels 2 3 or 4, which means what that the goal of my testing is to be able to find errors. So, to find errors, I have to first define what are my testing objectives in technical terms, and I have to be able to figure out how to effectively design test cases to be able meet or cover these test objectives.

So, in the course we will look at algorithms and techniques that will help you to formulate test objectives and design test cases that will help you to meet these objectives. So, the next module that we will be seeing, we will introduce the various terminologies that exist in testing and also clarify about what we would use is a part of this course.

Thank you.