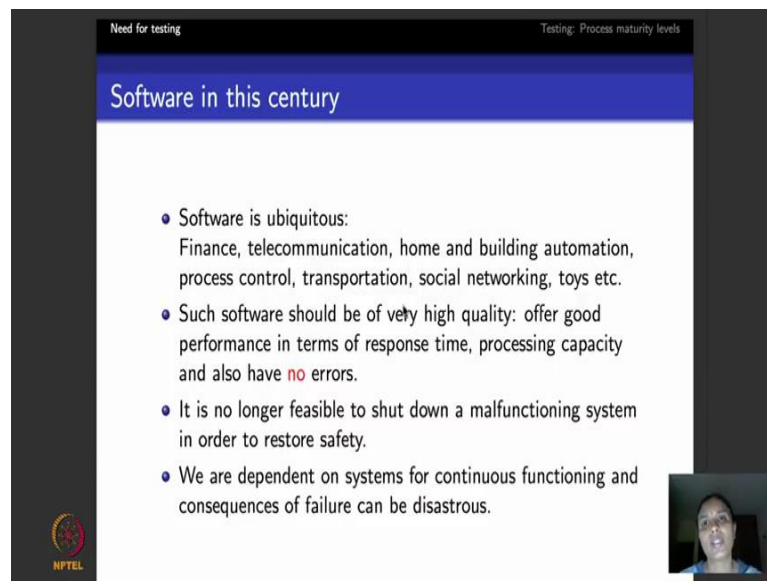


Software Testing
Prof. Meenakshi D'Souza
Department of Computer Science and Engineering
International Institute of Information Technology, Bangalore

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 extensive errors look like, and how to avoid these expensive errors by using testing. So, it is basically motivation module then we deal with how important testing is.

(Refer Slide Time: 00:45)



The slide is titled "Software in this century" and is part of a presentation on "Need for testing" and "Testing: Process maturity levels". It features a blue header and a white body with a list of four bullet points. The NPTEL logo is visible in the bottom left corner, and a small video inset of the speaker is in the bottom right corner.

- Software is ubiquitous:
Finance, telecommunication, home and building automation, process control, transportation, social networking, toys etc.
- Such software should be of very high quality: offer good performance in terms of response time, processing capacity and also have **no** errors.
- It is no longer feasible to shut down a malfunctioning system in order to restore safety.
- We are dependent on systems for continuous functioning and consequences of failure can be disastrous.

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

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

fly you 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. Slight 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)

The slide is titled "Some popular errors: Ariane 5" and is part of a presentation on "Need for testing" and "Testing: Process maturity levels". It lists three bullet points: "Ariane 5 rocket exploded in June 1996 36 seconds after it was launched.", "Reason: Software error", and "Exception occurred during conversion of a 64-bit floating point number into a 16-bit integer, backup software also failed due to same reason." The third bullet point is followed by "Rocket failed due to incorrect data transmission regarding altitude." Below the text are three small images showing the rocket launch and explosion. The NPTEL logo is in the bottom left corner, and a small video inset of a woman is in the bottom right corner.

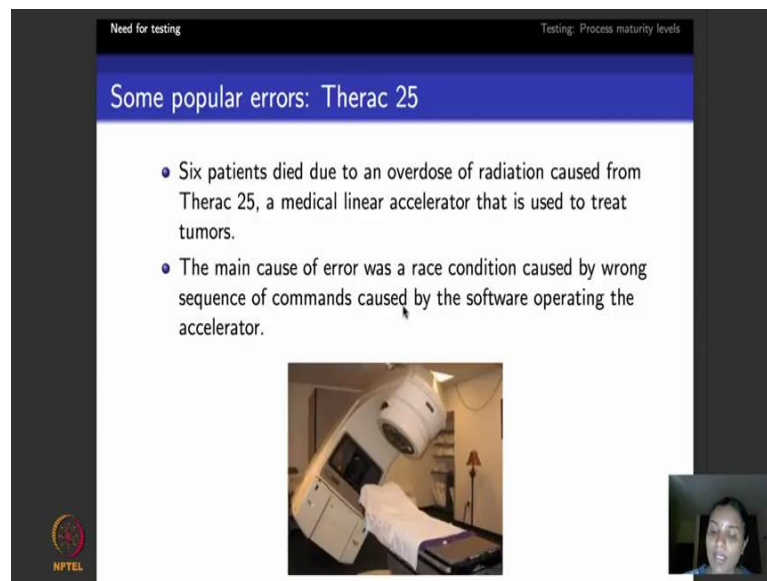
- Ariane 5 rocket exploded in June 1996 36 seconds after it was launched.
- Reason: Software error
- Exception occurred during conversion of a 64-bit floating point number into a 16-bit integer, backup software also failed due to same reason.
- Rocket failed due to incorrect data transmission regarding altitude.

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 notice 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, Benton 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)



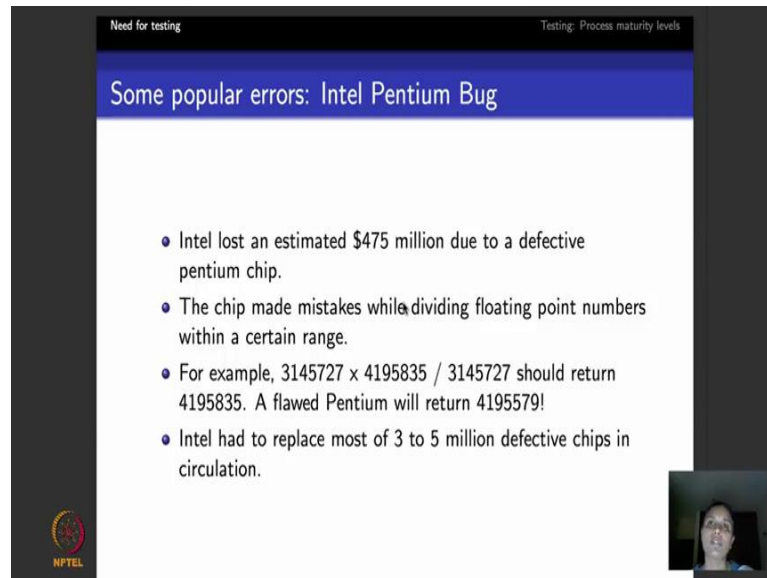
The slide is titled "Some popular errors: Therac 25" and is part of a presentation on testing. It contains the following text:

- Six patients died due to an overdose of radiation caused from Therac 25, a medical linear accelerator that is used to treat tumors.
- The main cause of error was a race condition caused by wrong sequence of commands caused by the software operating the accelerator.

The slide also features an image of the Therac 25 machine and a small video inset of a person in the bottom right corner. The NPTEL logo is visible in the bottom left corner.

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 ray's 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 patients lost their life because of an overdose of radiation.


(Refer Slide Time: 04:05)



Need for testing Testing: Process maturity levels

Some popular errors: Intel Pentium Bug

- Intel lost an estimated \$475 million due to a defective pentium chip.
- The chip made mistakes while dividing floating point numbers within a certain range.
- For example, $3145727 \times 4195835 / 3145727$ should return 4195835. A flawed Pentium will return 4195579!
- Intel had to replace most of 3 to 5 million defective chips in circulation.

NPTEL 

So, the next kind of example that I would like to discuss with you it is software that is expensive that caused somebody 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)

The slide is titled "Some more expensive errors!" and is part of a presentation on "Need for testing" and "Testing: Process maturity levels". It features a list of bullet points:

- NIST report: The Economic Impacts of Inadequate Infrastructure for Software Testing (2002):
 - Inadequate software testing costs the US alone between \$ 22 and \$ 59 billion annually.
 - Better approaches could cut this amount in half.
- Huge losses due to web application failures.
 - Financial services : \$ 6.5 million per hour (just in USA!)
 - Credit card sales applications : \$ 2.4 million per hour (in USA)
 - In Dec 2006, amazon.coms BOGO offer turned into a double discount
- 2007: Symantec says that most security vulnerabilities are due to faulty software.

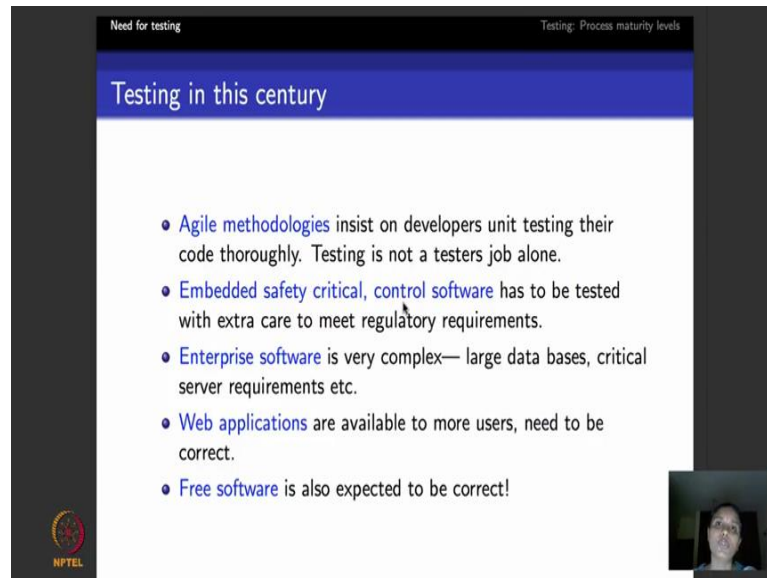
The slide also includes an NPTEL logo in the bottom left corner and a small video inset of a speaker in the bottom right corner.

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 right. 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 right. The other popular categories of losses are due to web applications which we use always all that I am write. 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, right.

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 transaction then our online wallet us and so on mainly occur no because of cryptographic errors, they occur because of software errors, right.

(Refer Slide Time: 06:21)



The slide is titled "Testing in this century" and is part of a presentation on "Need for testing" and "Testing: Process maturity levels". It features a list of five bullet points:

- Agile methodologies insist on developers unit testing their code thoroughly. Testing is not a testers job alone.
- Embedded safety critical, control software has to be tested with extra care to meet regulatory requirements.
- Enterprise software is very complex— large data bases, critical server requirements etc.
- Web applications are available to more users, need to be correct.
- Free software is also expected to be correct!

The slide also includes the NPTEL logo in the bottom left corner and a small video feed of a person in the bottom right corner.

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 self 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, right.

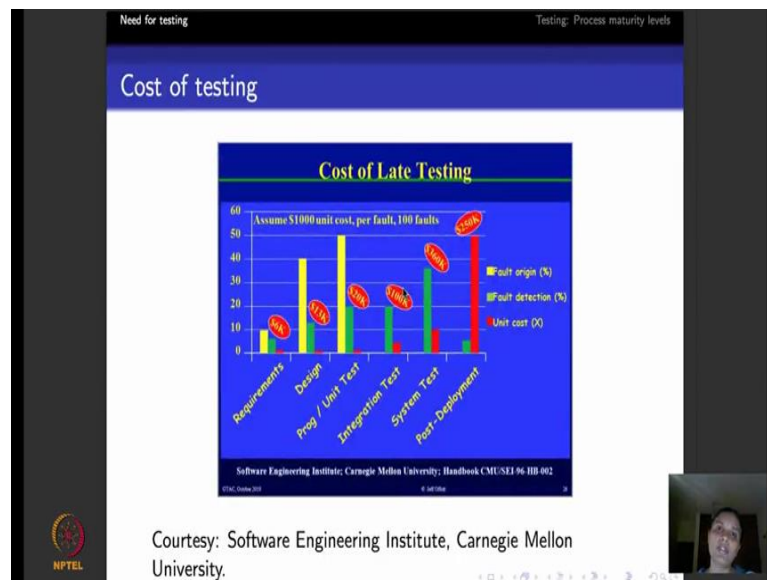
Now, let us look at the various kinds of software that we will deal with right. If you consider a software that typically runs in your cars that helps you to do breaking automatically, or does a cruise control or a software that runs in your aeroplanes, such software is what is called embedded control 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 why 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 and 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 right. Just because it is available for free you are not willing to accept the fact this software could have errors right. 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 that right. 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)

Need for testing

Testing: Process maturity levels

Testing: Facts and Myths

- Fact: Testing can be used to find errors in software, cannot be used to show that a software is correct.
- Fact: Testing cannot be replaced by software reviews, inspections, quality audits etc.
- Fact: Testing cannot be fully automated, needs human intervention.
- Myth: It is wrong to say that "My code is correct and doesn't need to be tested".

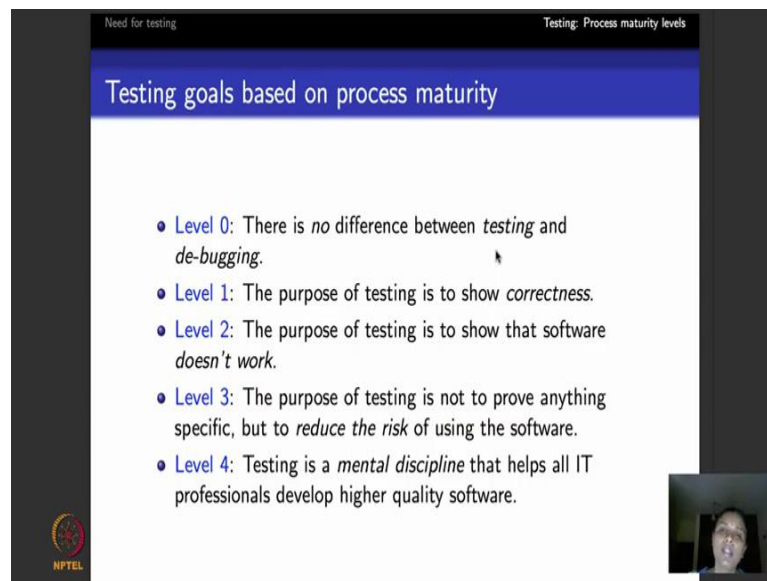
NPTEL

So, what are the facts and myths about testing? There is a popular saying by (Refer Time: 09:58) 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 I have written the code, I have debugged it well and my quality auditor my (Refer Time: 10:30) 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 codes is about needs a human to be able to do it right.

(Refer Slide Time: 11:18)



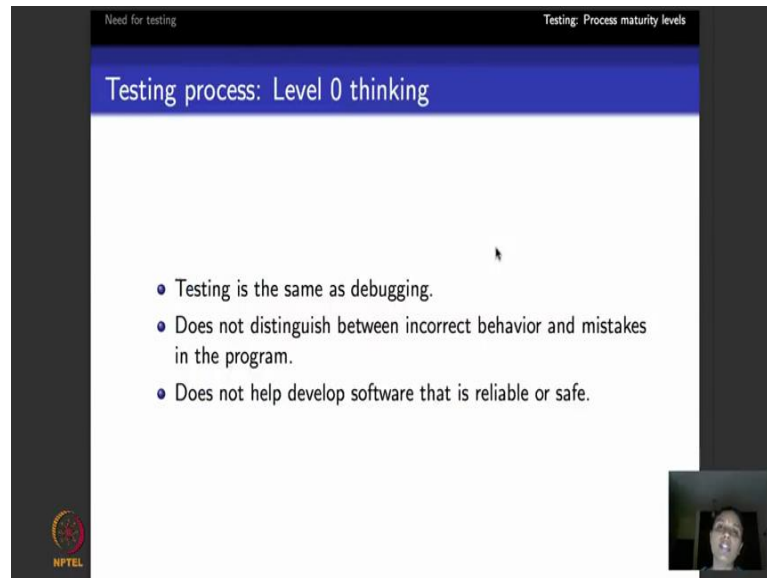
The slide is titled "Testing goals based on process maturity" and is part of a presentation on "Testing: Process maturity levels". It lists five levels of testing maturity:

- **Level 0:** There is *no* difference between *testing* and *de-bugging*.
- **Level 1:** The purpose of testing is to show *correctness*.
- **Level 2:** The purpose of testing is to show that software *doesn't work*.
- **Level 3:** The purpose of testing is not to prove anything specific, but to *reduce the risk* of using the software.
- **Level 4:** Testing is a *mental discipline* that helps all IT professionals develop higher quality software.

The slide also features an NPTEL logo in the bottom left corner and a small video inset of a person in the bottom right corner.

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)



Need for testing

Testing: Process maturity levels

Testing process: Level 0 thinking

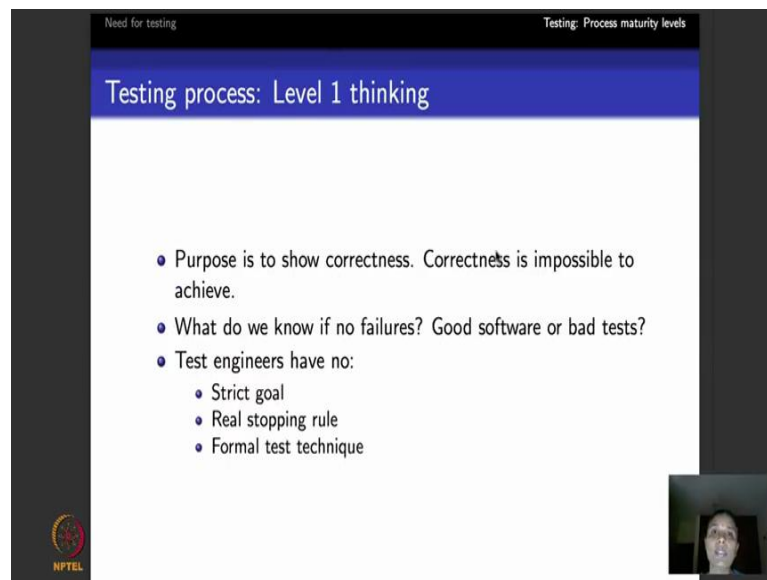
- Testing is the same as debugging.
- Does not distinguish between incorrect behavior and mistakes in the program.
- Does not help develop software that is reliable or safe.

NPTEL

Video inset showing a person speaking.

So, what is 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)



Need for testing

Testing: Process maturity levels

Testing process: Level 1 thinking

- Purpose is to show correctness. Correctness is impossible to achieve.
- What do we know if no failures? Good software or bad tests?
- Test engineers have no:
 - Strict goal
 - Real stopping rule
 - Formal test technique

NPTEL

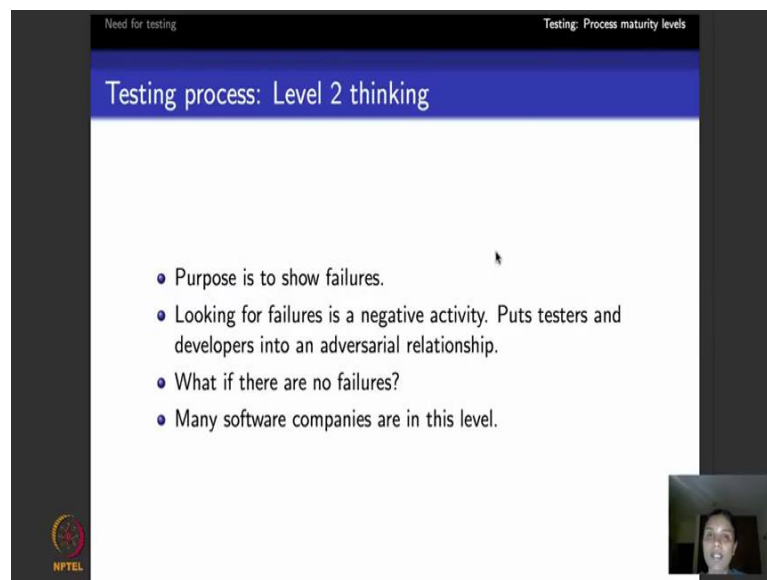
Video inset showing a person speaking.

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 testes 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 $2^{32} - 1$ to $2^{32} + 1$, 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)



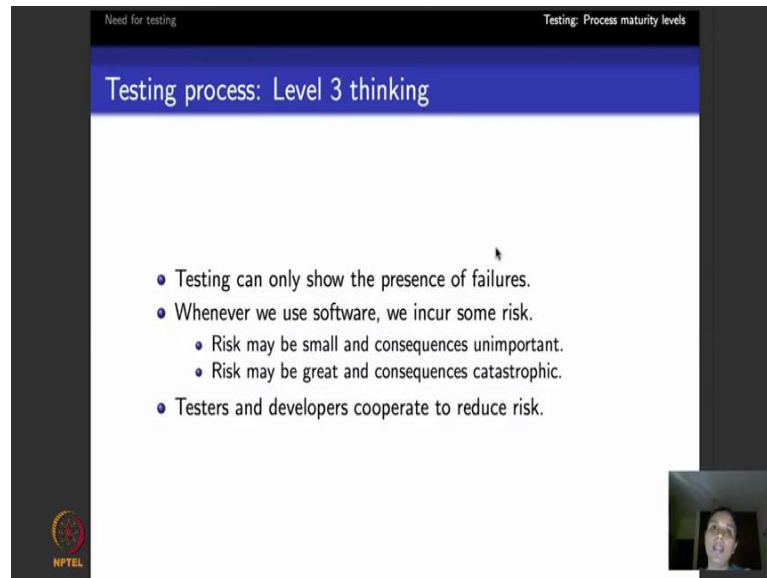
The slide is titled "Testing process: Level 2 thinking" and is part of a presentation on "Testing: Process maturity levels". It contains the following bullet points:

- Purpose is to show failures.
- Looking for failures is a negative activity. Puts testers and developers into an adversarial relationship.
- What if there are no failures?
- Many software companies are in this level.

The slide also features an NPTEL logo in the bottom left corner and a small video inset in the bottom right corner showing a person speaking.

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 in organizations that are typically at this level there is a lot of tiff between developers and testers, they belong to different teams and then they 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 right.

(Refer Slide Time: 13:28)




Need for testing

Testing: Process maturity levels

Testing process: Level 3 thinking

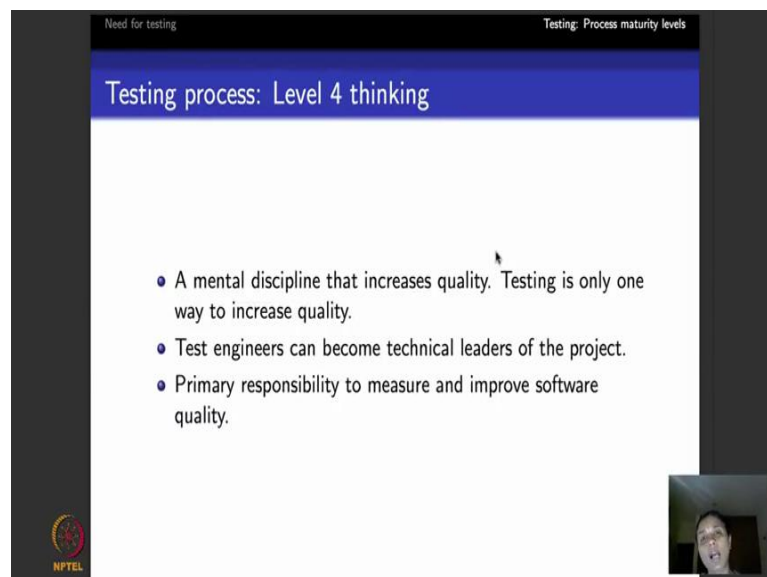
- Testing can only show the presence of failures.
- Whenever we use software, we incur some risk.
 - Risk may be small and consequences unimportant.
 - Risk may be great and consequences catastrophic.
- Testers and developers cooperate to reduce risk.

NPTEL



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)




Need for testing

Testing: Process maturity levels

Testing process: Level 4 thinking

- A mental discipline that increases quality. Testing is only one way to increase quality.
- Test engineers can become technical leaders of the project.
- Primary responsibility to measure and improve software quality.

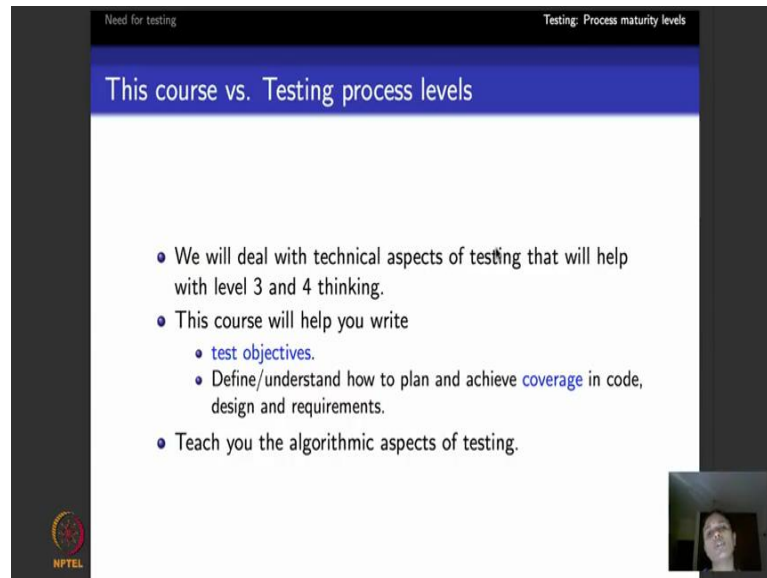
NPTEL



Level 4 thinking is very 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 right. 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)



The slide is titled "This course vs. Testing process levels" and is part of a presentation on testing. It features a blue header bar with the title. The main content is a list of bullet points. In the bottom right corner, there is a small video inset showing a person's face. The slide also includes a logo for NPTEL in the bottom left corner.

- We will deal with technical aspects of testing that will help with level 3 and 4 thinking.
- This course will help you write
 - test objectives.
 - Define/understand how to plan and achieve coverage in code, design and requirements.
- Teach you the algorithmic aspects of testing.

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.