

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

Lecture – 57
Software Reliability – I

Welcome to this lecture. In the last lecture we had looked at some very basic aspects of Software Reliability. We had said that quality and reliability are closely related and the software quality to large extent is determined by it's reliability.

If a software is of not if poor reliability we cannot say that it is a good quality and with that intention we started discussing about some very basic aspects of software reliability. Like what do we mean by Reliability? How do we formally define reliability? And then we were discussing about why software reliability is much difficult to measure why the software reliability study is very different from hardware reliability study and so on.

(Refer Slide Time: 01:26)



Now, let us continue from that point onwards. We will first briefly look at why software reliability measurement is a much harder problem than hardware reliability measurement or measurement of the reliability of any other equipment, structure and so on. And then we will define some metrics for software reliability measurement and then we will say that to meaningfully estimate software reliability. We have to use the two approaches

which is reliability growth modeling and statistical testing and we will see the pros and cons of both these approaches.

(Refer Slide Time: 02:14)

Major Problems in Reliability Measurements (1)

- All errors do not cause failures at the same frequency and severity.
- Removing errors from parts of software which are rarely used: Makes little difference to the perceived reliability.

→ **Estimating latent errors alone not enough to determine reliability.**

The slide includes a diagram of a yellow circle labeled 'Software' with a red dot and an arrow pointing to a smaller circle labeled 'Core'. At the bottom, there is a video feed of a man and logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES.

We have been saying that measurement of software reliability is a hard problem. There are many factors which contribute to making this hard problem, the first issue is that a software will have many errors and one would expect that if we can estimate the number of errors we can derive the reliability from that. But unfortunately all errors do not cause failures at the same frequency and severity. Actually there is a very very wide difference with the frequency with which different errors called system failure and also the severity of the system failure.

There are some errors which rarely show may be after a million hours of uses may show up once. Whereas, another error may show up even in the first minute of users, the error which is occurs very rarely if we remove that error it will make a very little difference to the reliability of the software the perceived reliability. But the one which appears again and again in the form of failures that would result, if we can eliminate that error it will result in significant improvement of the reliability of the software.

From this argument it is clear that even if we are somehow been able to estimate a number of errors remaining in software, that is not enough to estimate the reliability of the software. If this is the software then as the code executes some code some part of the code that executes almost all the time, the other part does not execute as frequently and

this part marked in the red is called as the core of the program those statements are executed again and again.

(Refer Slide Time: 05:00)

The 90-10 Rule

- Experiments from analysis of behavior of a large number of programs:
 - 90% of the total execution time is spent in executing only 10% of the instructions in the program.
- The most used 10% instructions:
 - called the core of the program.

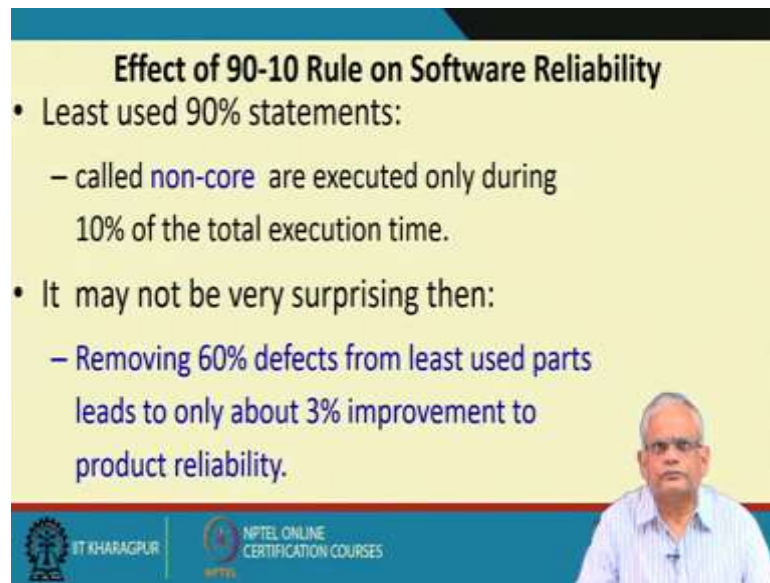
Software
Core

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

The 90-10 rule says that 90 percent of the execution time is spent only in 10 percent of the instructions of the program, the rest 90 percent of the instructions execute only 10 percent of the time and 10 percent of instruction execute 90 percent of the time.

And therefore, if we remove a error appearing in the core part that will result in a significant improvement of reliability. Whereas the one which is very rarely used would make hardly any difference to the reliability.

(Refer Slide Time: 05:46)



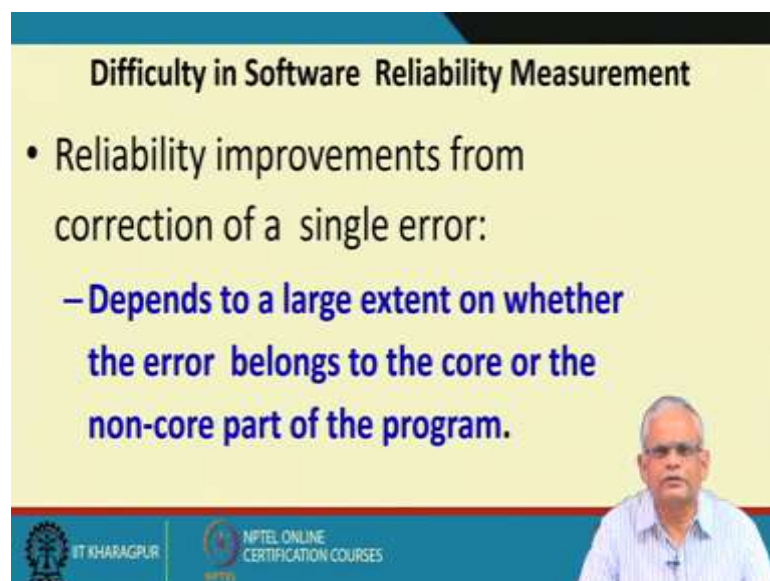
Effect of 90-10 Rule on Software Reliability

- Least used 90% statements:
 - called **non-core** are executed only during 10% of the total execution time.
- It may not be very surprising then:
 - Removing **60% defects from least used parts** leads to only about **3% improvement to product reliability.**

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Some studies suggest that if a software has let us say thousand bugs and if we remove six hundred of those bugs which are there in the noncore part. It results only 3 percent improvement to the reliability, very very surprising result, that removing almost more than half the bugs results in only 3 percent improvement. Because those bugs happen to be in the noncore part and this issue makes reliability estimation from the number of latent error estimation not very feasible.

(Refer Slide Time: 06:47)



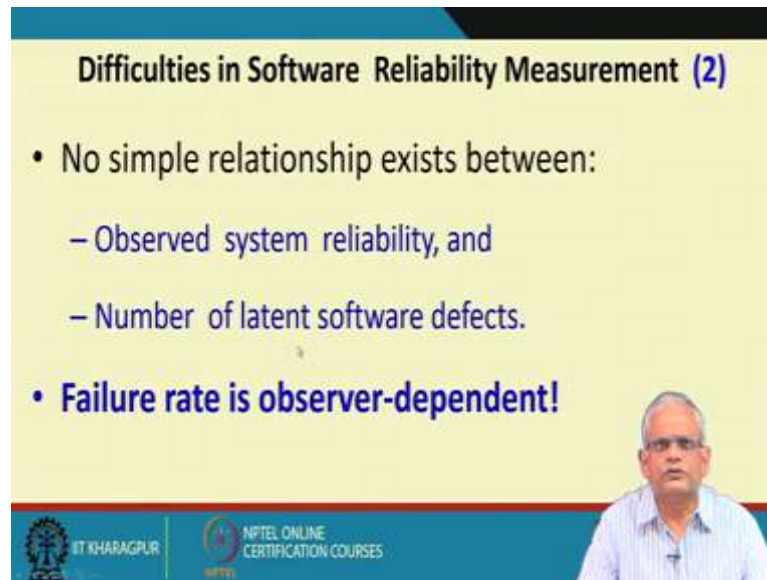
Difficulty in Software Reliability Measurement

- Reliability improvements from correction of a single error:
 - **Depends to a large extent on whether the error belongs to the core or the non-core part of the program.**

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

The reliability improvement from correction of a single error depends to a large extent whether the error belongs to the core part or the noncore part of the program, this is one issue which we need to keep in mind when we think of software reliability.

(Refer Slide Time: 07:11)



Difficulties in Software Reliability Measurement (2)

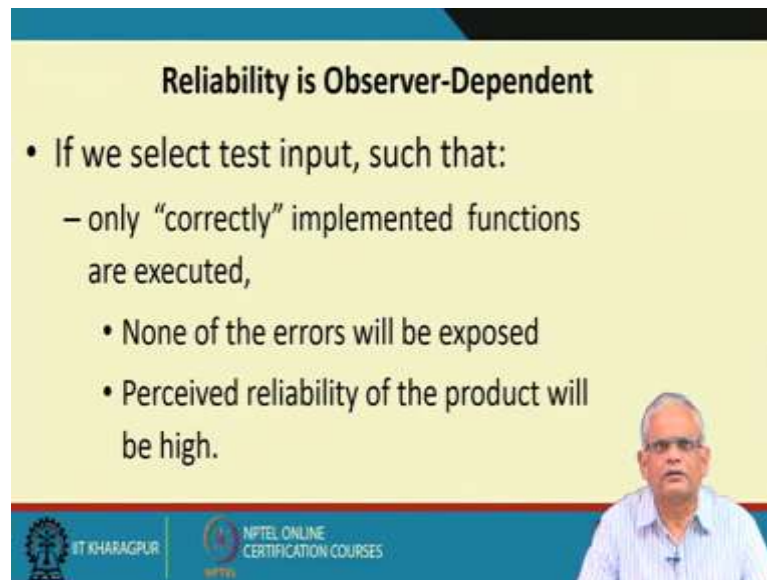
- No simple relationship exists between:
 - Observed system reliability, and
 - Number of latent software defects.
- **Failure rate is observer-dependent!**

The slide features a video inset of a man in a light blue shirt speaking. At the bottom, there are logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES.

Another issue so this is still in the first issue, because the frequency with which a error causes failure depends on the part of the program at which the error lies. Therefore, the system reliability the observed system reliability and the number of latent software defects, there is no simple relationship otherwise you can derive or empirically propose a expression in which you should be able to input the number of latent software defects and get the reliability. But this is the main reason why we cannot do that.

Now, let us look at the second issue, the second issue is the failure rate is observer dependent, a software which appears to be very reliable to one person may appear to be very unreliable to another person. Now, let us investigate the reason behind this.

(Refer Slide Time: 08:33)



Reliability is Observer-Dependent

- If we select test input, such that:
 - only “correctly” implemented functions are executed,
 - None of the errors will be exposed
 - Perceived reliability of the product will be high.

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

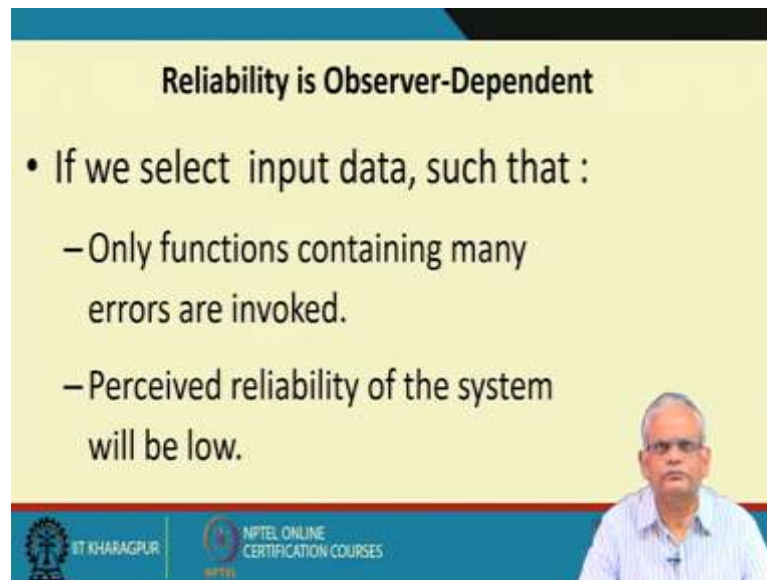
The slide features a yellow background with a blue header and footer. A small video feed of a man in a light blue shirt is visible in the bottom right corner of the slide area.

Let us say one user selects inputs or executes the software. So that only the correct functionalities are executed a software has hundreds of functionalities and let us say each user is interested in executing only part of the functionality. Let us say a library software the students or the members are interested to issue the book, return book, query book and so on. The librarian would be executing the functions create member, delete member create book, collect statistics and so on.

The accounts department would again be using the same software, but executing very different functionality. Like what is the income from fee collection membership collection how much is from the fine collection how much grant is received how much books have been procured this year and so on.

So, every user every type of user use different functionalities of the same software. Now, let us say one type of user they are executing functionalities which are correctly implemented functions and they find that all the functions they execute work perfectly all right and they would give the opinion that the software is really good highly reliable. On the other hand for another category of users almost every function they execute displays failure error and so on and they would form the opinion that the software is of poor quality it is unreliable.

(Refer Slide Time: 10:36)



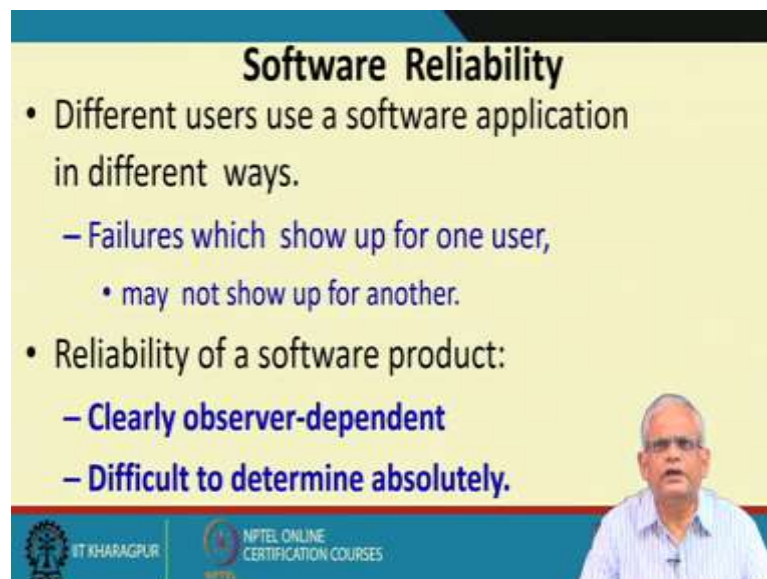
Reliability is Observer-Dependent

- If we select input data, such that :
 - Only functions containing many errors are invoked.
 - Perceived reliability of the system will be low.

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

And therefore, for the same software two different users can have two different opinion about it's reliability. The opinion of a user about the reliability of a software we call as the perceived reliability. So, the perceived reliability can be low for one group of users and the perceived reliability can be high for another group of users.

(Refer Slide Time: 11:08)



Software Reliability

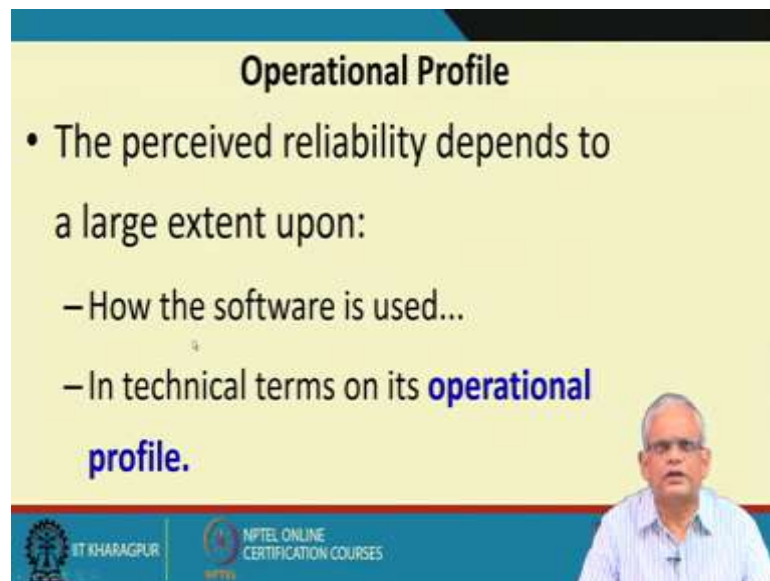
- Different users use a software application in different ways.
 - Failures which show up for one user,
 - may not show up for another.
- Reliability of a software product:
 - **Clearly observer-dependent**
 - **Difficult to determine absolutely.**

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

We are just mentioning this point the different users of a software application, they use the software in different ways they are interested in different functionalities and therefore the failure which shows for another one user may not use so for another user.

So, different users can come up with different reliability numbers for the same software and therefore it is clearly observer dependent. But again if we have to give one absolute value how do we do it, which user will take into account. Because we will get different numbers for different users, we will see how to handle that problem, but then it makes reliability estimation complicated because the reliability has different observers give different reliability numbers to the same software. And it makes it difficult to give a absolute reliability number a single reliability number to the software.

(Refer Slide Time: 12:44)



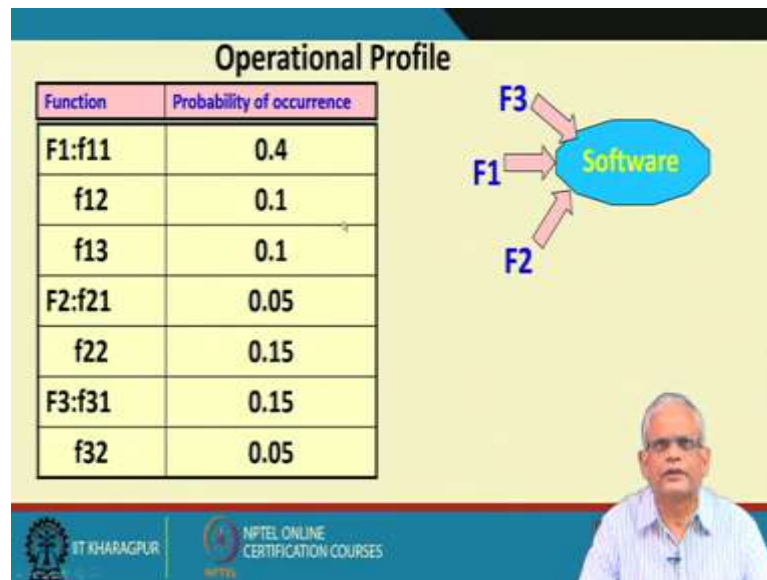
Operational Profile

- The perceived reliability depends to a large extent upon:
 - How the software is used...
 - In technical terms on its **operational profile.**

The slide features a yellow background with a blue header and footer. A small inset video of a man in a light blue shirt is visible in the bottom right corner. The footer contains the logos for IIT Kharagpur and NPTEL Online Certification Courses.

The way it is done that is to give a single reliability number is to consider the operational profile of the software. And the if you have to give a single number we will have to give it based on it is operational profile. Since the perceived reliability depends to a large extent on how the software is used or in other words it is operational profile. Let us first define what is operational profile and then we will see how to give a single reliability number to a software.

(Refer Slide Time: 13:33)



Let us say a software has many functionalities F1, F2, F3 etcetera and also each of these functionalities will have different scenarios of operation. Let us say this is an ATM software, bank ATM and then one function is withdraw cash, one is the main line scenario that is, go there do everything correctly give password insert the amount to be withdrawn and cash is dispensed come out.

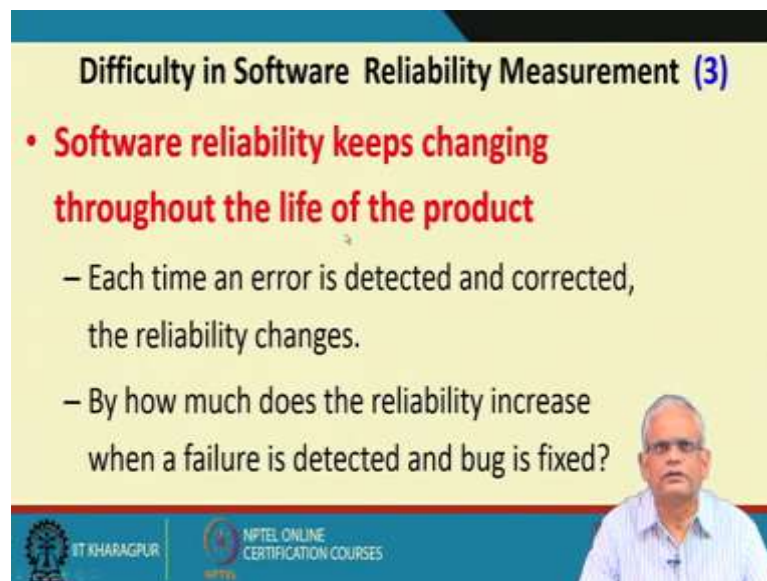
The second scenario is that the password may be incorrect or the card may be rejected or may be that the account does not have enough balance. So, that is another scenario one more scenario can be that the ATM does not have enough money to dispense and so on.

So, the same functionalities can have different scenarios of operation. We can observe the execution of these different functions by the users, we can take a log of that as the different users use a log gets collected and we analyze which function and which scenario gets executed how many times. So, that is function and the probability of occurrence we might have these three functions F1, F2, F3 and we may determine that one scenario the first scenario F11 gets executed 40 percent of the time, F1 2 the second scenario of the F1 function gets executed 10 percent of the time and F1 3 the third scenario for F1 get executed 10 percent of time.

The first scenario of F2 gets executed 5 percent of time and the second scenario gets 15 percent of time. The first scenario third use case the third function gets executed 15 percent of time and the second scenario of the third function gets executed 5 percent of

time. And this we call as it is operational profile of the software. Let me just repeat here that the operational profile of a software is constructed by observing the way the software is used by various users over a long enough period of time and then finding what is the rate at which the different functions get executed. The probability that the first scenario of F1 gets executed is forty percent and so on, this we call as the operational profile of the software.

(Refer Slide Time: 16:42)



Difficulty in Software Reliability Measurement (3)

- **Software reliability keeps changing throughout the life of the product**
 - Each time an error is detected and corrected, the reliability changes.
 - By how much does the reliability increase when a failure is detected and bug is fixed?

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

And once we have the operational profile that is the average system users of all types of users and therefore based on the operational profile. We can give a single reliability number to a software and that we will discuss a little later how to go about giving a single reliability number to a software based on it's operational profile. And we call that a statistical testing.

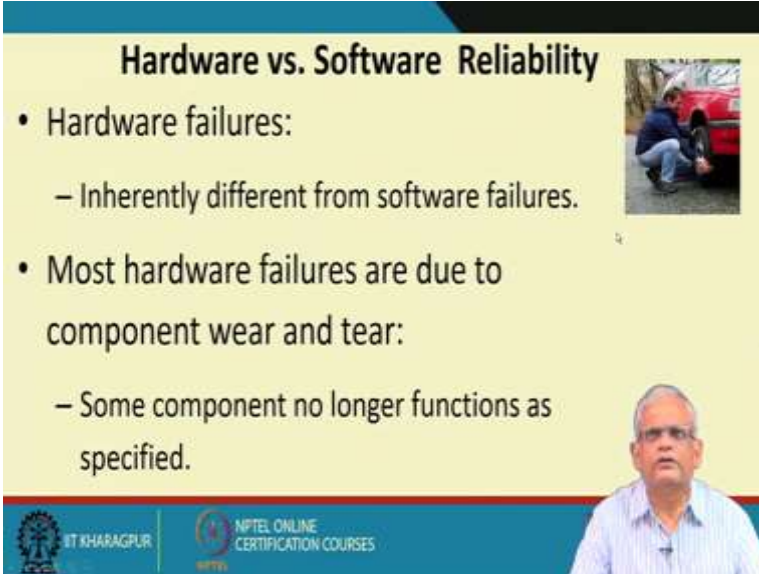
Now, let us look at the third issue which makes reliability measurement difficult, this problem is that the reliability of a software keeps on changing throughout it is life cycle. Because each time there is a failure, the software is debugged, the error is detected and corrected and therefore the same type of failure will not occur.

But then the improvement of reliability is not fixed some error fixes may cause large improvement of reliability, some may cause less improvement in reliability and so on. But typically as the software gets used, the errors are expressed in failures and these are corrected the reliability keeps on improving for the software. Unlike hardware where the

reliability more or less remains constant, in software the reliability changes and typically improves.

And therefore, becomes very difficult to give a reliability number to a software, because if we give it today, then after a year the same number will not hold or may be after a few days the same number will not hold. So, it becomes difficult to give a number reliability number, reliability estimation value which can remain valid for sometime.

(Refer Slide Time: 19:22)



Hardware vs. Software Reliability

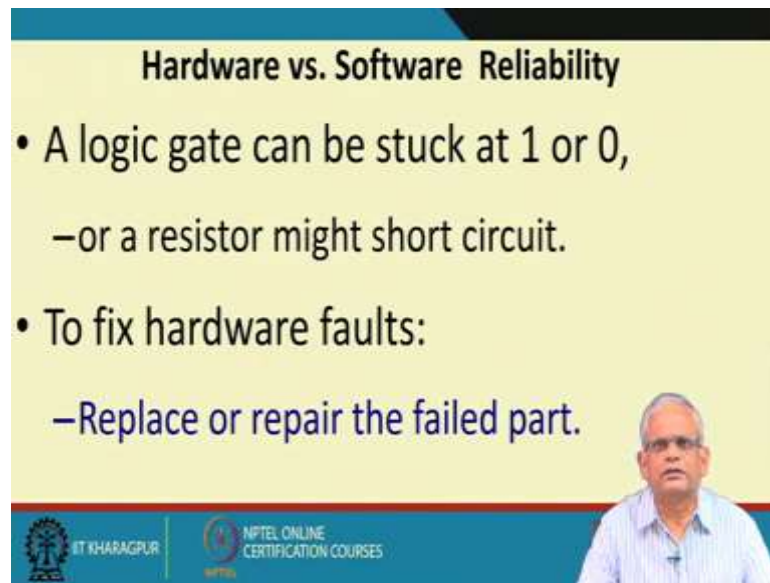
- Hardware failures:
 - Inherently different from software failures.
- Most hardware failures are due to component wear and tear:
 - Some component no longer functions as specified.

The slide includes a small image of a person working on a car and a video inset of a man speaking. Logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES are visible at the bottom.

If we compare hardware and software reliability there are some issues which are very different in hardware and software. The first thing is that the type of failure that occur in hardware is very different very different from the failure that occurs in software. In hardware the failures occur due to component wear and tear, if it is a automobile the failures that typically occur are let us say the tyre punctured the tyre worn out replace the tyre the battery become old, replace the battery and so on.

So, if we analyze hardware failures like a car or a mixer grinder, we find that the failure is due to use of the system and some components, they wear and they break and the failures are largely due to component breaking due to use or aging. But in software there is no such thing as wear and tear, here this is a car the wheel has worn out punctured and it can be replaced and the failure here is due to wear and tear. Whereas, in software there is no wear and tear, this is one major difference between software failures and hardware failures.

(Refer Slide Time: 21:36)



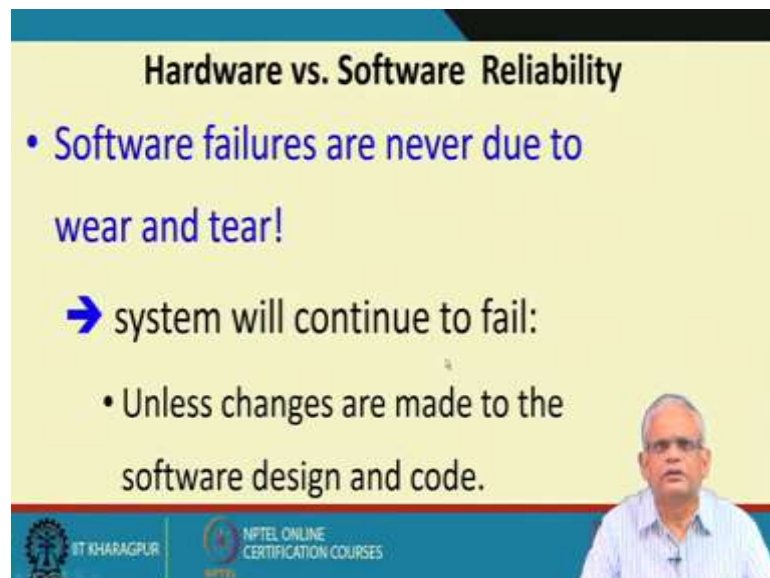
Hardware vs. Software Reliability

- A logic gate can be stuck at 1 or 0,
–or a resistor might short circuit.
- To fix hardware faults:
–Replace or repair the failed part.

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

If it is a logic circuit a logic gate can become short circuited or open circuit and therefore it may be 1 or 0 in all types of hardware. If there is a failure typically find out which component is failing and replace that part and then the reliability is back to what it was. A reliability estimation can be done for a hardware system, a number can be given and there can be failures. But then on failure we replace a part and the system is back to the previous reliability.

(Refer Slide Time: 22:30)



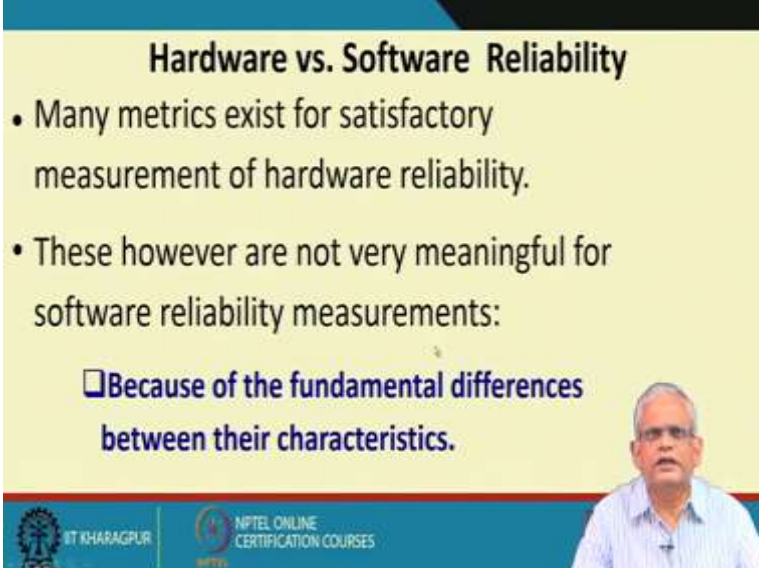
Hardware vs. Software Reliability

- Software failures are never due to wear and tear!
→ system will continue to fail:
 - Unless changes are made to the software design and code.

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

But software the failures are not due to wear and tear, the failures are due to bugs and here the only way to correct the failure is to remove the bug. And once we change the code and remove the bug the reliability changes, the system continuous to fail unless changes are made to the design and code.

(Refer Slide Time: 23:03)



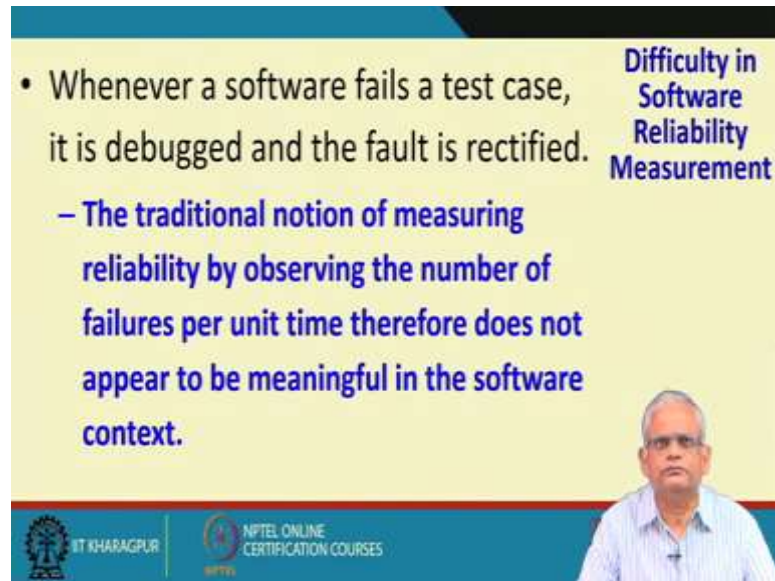
Hardware vs. Software Reliability

- Many metrics exist for satisfactory measurement of hardware reliability.
- These however are not very meaningful for software reliability measurements:
 - **Because of the fundamental differences between their characteristics.**

The slide features a speaker in the bottom right corner and logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES at the bottom.

And once we change the design the reliability changes typically the reliability improves. Now, the metrics that are used to measure hardware reliability, they are basically based on observing the number of failures over a period of time long enough time and then noting the number of failures and giving a reliability value. But for software these metrics will not be very meaningful because, the failure types are very different here the each failure results in reliability improvement in software. Whereas, the hardware the reliability is maintained.

(Refer Slide Time: 23:59)



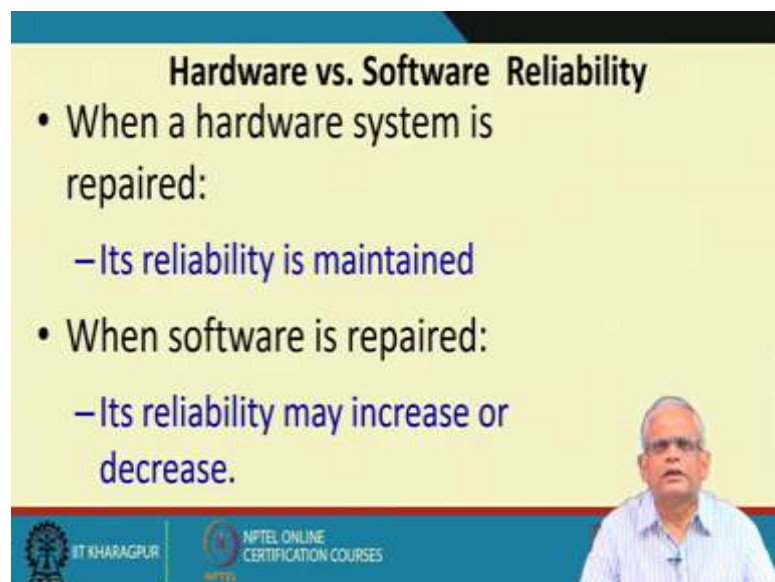
Difficulty in Software Reliability Measurement

- Whenever a software fails a test case, it is debugged and the fault is rectified.
 - The traditional notion of measuring reliability by observing the number of failures per unit time therefore does not appear to be meaningful in the software context.

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Whenever a software fails at test case there is a failure it is debugged the fault is rectified and therefore the traditional notion of measuring reliability by observing the number of failures per unit time does not appear to be meaningful in the software context.

(Refer Slide Time: 24:23)



Hardware vs. Software Reliability

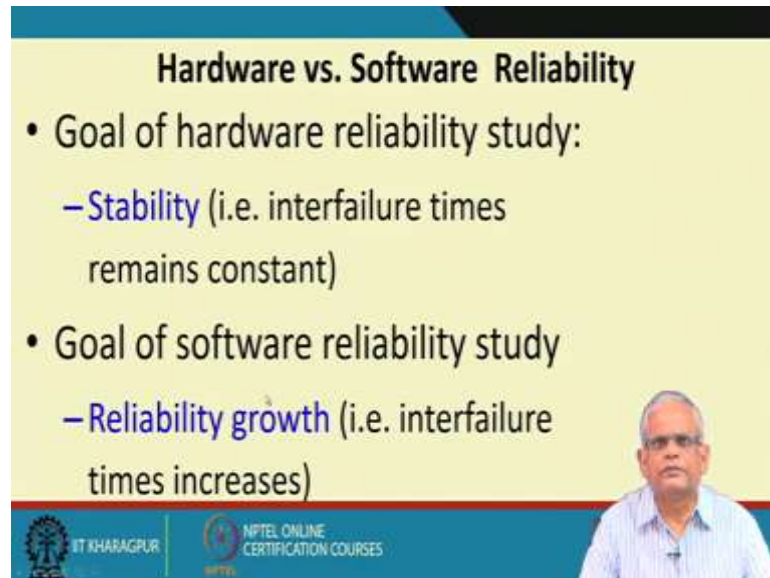
- When a hardware system is repaired:
 - Its reliability is maintained
- When software is repaired:
 - Its reliability may increase or decrease.

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

In hardware system once there is a failure, the component is replaced and the reliability is maintained. On the other hand, when software is repaired it is reliability may increase in that we have removed a bug or it may decrease also, because one fix error fix may create new bugs, let us say we fixed one bug. And let us say four five new bugs got

introduced because that error fix was not proper it fixed that bug, but then there were some other bugs which resulted due to that bug fix.

(Refer Slide Time: 25:15)



The slide is titled "Hardware vs. Software Reliability" and is presented on a yellow background. It contains two main bullet points. The first bullet point is "Goal of hardware reliability study:" followed by a sub-point "– Stability (i.e. interfailure times remains constant)". The second bullet point is "Goal of software reliability study" followed by a sub-point "– Reliability growth (i.e. interfailure times increases)". In the bottom right corner of the slide, there is a small video inset of a man with glasses and a light blue shirt. At the bottom of the slide, there are logos for "IIT KHARAGPUR" and "NPTEL ONLINE CERTIFICATION COURSES".

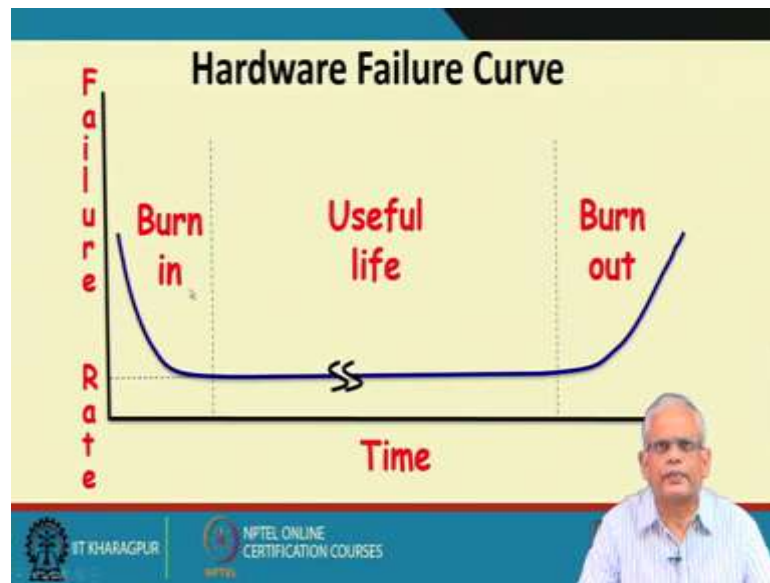
- Goal of hardware reliability study:
 - Stability (i.e. interfailure times remains constant)
- Goal of software reliability study
 - Reliability growth (i.e. interfailure times increases)

So, the we can say that the goal of hardware reliability study is stability that is the reliability is maintained the inter failure times remain constant, once the failure conditions are removed.

On the other hand the goal of software reliability study is reliability growth that is typically the inter failure times decreases the inter failure times decrease. I am sorry the there is a reliability growth the inter failure times increases that is, it does not fail as often because bugs are being removed. So, there is a reliability growth or inter failure times increases.

So, the study of hardware reliability stability, reliability stability, whereas the goal of software reliability is reliability growth the inter failure times increases. Whereas, in hardware the inter failure times remain constant.

(Refer Slide Time: 26:24)



So, we can see behavior of hardware and software failure in a diagram called as the bathtub curve. We are almost at the end of this lecture we will continue from this point in the next lecture.

Thank you.