

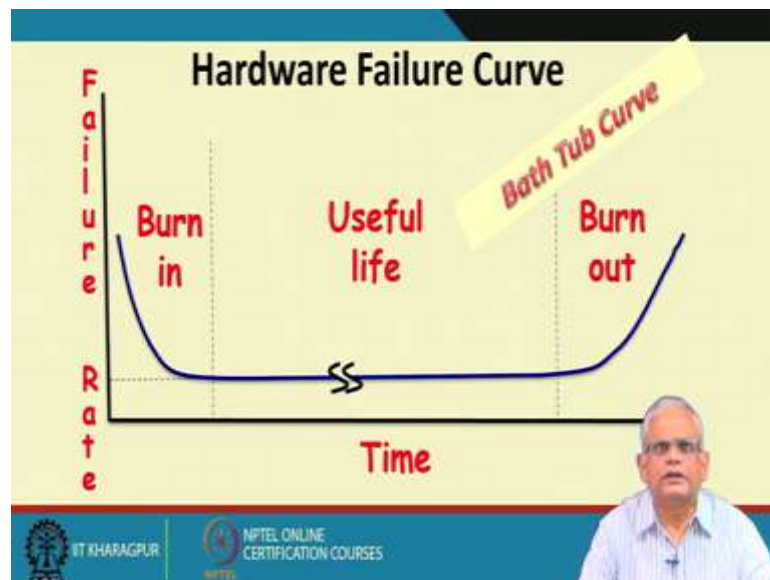
Software Project Management
Prof. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 58
Software Reliability – II

Welcome to this lecture. In the last lecture we were discussing the difference between hardware reliability and Software Reliability. We said that hardware reliability measurement is a relatively simple problem, because the failures there are due to wear and tear of components. And, once we replace the broken component, the reliability comes back to its previous reliability or the reliability is maintained.

Whereas, in software the failures are not due to wear and tear, these are due to bugs and to correct the failure it is debugged and the error is corrected. And therefore, the reliability increases or the inter-failure times increases. And therefore, the reliability keeps on changing in case of software whereas, in case of hardware the reliability is maintained.

(Refer Slide Time: 01:40)



Now, let us look at this issue a little deeper. This is a typical behavior of a hardware system, here we have plotted the failure rate that is number of failures per unit time with the time of users. For a automobile or something it may be let us say for a period of 30 years or something and for a simpler system it may be 1 year and so on. Now, every

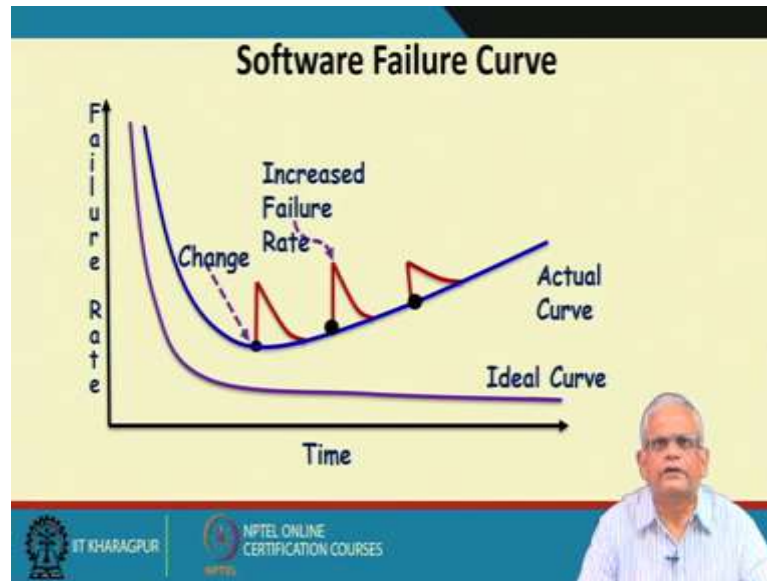
hardware system it shows this kind of behavior that initially the failure rate is very high and then with time it decreases. It may be automobile, it may be a mixer grinder, it may be a refrigerator, it may be a television set.

Any hardware system initially shows very high failure rate and the failure rate keeps on decreasing and then for a long time the failure rate is maintained. The inter-failure times remain constant over a long period of time that we have shown the symbol. And, this is called as the lifetime of the system; when the failure rate starts to increase. And, this is called as the bathtub curve, it is a typical model of any hardware system reliability. Initially the weak systems, the weak components each hardware system is made up of many components. Initially the weaker components they fail and may be the interfaces among components they fail recurring interfaces.

So, this is called as the burn in where the weaker components and weaker interfaces among components they fail and then they are repaired and reliability becomes high, stable and maintained in this rate for a long time. Typically, the initial part is covered by the manufacturer's warranty otherwise the people will not buy the hardware system because, it is showing a very high failure rate.

Typically, all good manufacturers cover this part by the manufacturer's warranty because of the burn in. The reliability is maintained and at the end of the lifetime, this point is the lifetime the failure rate starts increasing. And, here most of the components are worn out and the major components start failing.

(Refer Slide Time: 05:08)



In contrast to a hardware failure curve, the software failure curve appears like this. It should actually ideally it should appear like this violet curve, that as the time passes. The different problems or failures are reported and these are corrected and typically the failure rate is high of the for the software, any software. Once, these are released and these are fixed, as they are fixed the reliability improves and the failure rate comes down and then it slowly becomes better and better. Initially, there is a steep fall because the frequently occurring failures or the failures in the core part of the software get noticed and they are repaired.

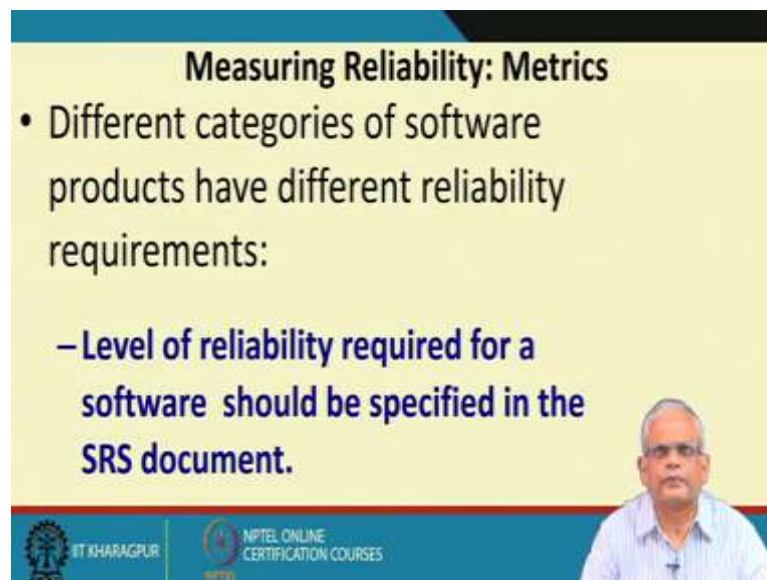
And, each of those repair causes a huge improvement in the reliability and later slowly the more esoteric or rarely used parts of the software errors are found and those are fixed and they little improve the reliability. But, actually the curve is not like this, the actual software failure curve is that initially there is a rapid increase in reliability or decrease in failure rate. But, then this is not smooth curve throughout and the curve I have just shown few of this once here, but throughout this there are glitches like this. Each time there is a failure reported and these are corrected, there is a jump here because of a fix causing other bugs to appear.

Of course, sometimes the fix may be very proper, but sometimes a fix, bug fix may cause new bugs to arise and therefore, the reliability decreases after a fix. And, then after sometime the reliability decreases; initially the reliability increases or the failure rate

comes down and then there is temporary glitches, these are due to the bug fix creating new bugs. But just see the trend here, the unmistakable trend here is that the reliability is decreasing over time. Why is that? The reason is that because of many bug fixes slowly the structure of the software becomes poor. And therefore, the reliability becomes poor, many ad hoc fixes degrade the software structure and therefore, it gets poorer reliability.

We can see that the failure curve for software is very different from the bathtub curve for hardware systems. In other words the behavior, the reliability behavior of hardware and software are very different.

(Refer Slide Time: 09:21)



Measuring Reliability: Metrics

- Different categories of software products have different reliability requirements:
 - **Level of reliability required for a software should be specified in the SRS document.**


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

Now, let us see how do we measure reliability because, the user should be interested. They may specify the level of reliability for a software, typically for safety critical software they would definitely give a reliability figure. And, even the other users they are concerned about the down time, the frequency with which failures appear and so on and they may specify a failure rate for a software, they would like to install on their systems. But, using what metrics? Let us look at that issue that what are the metrics for measuring reliability.

(Refer Slide Time: 10:12)

Reliability Metrics

- A good reliability measure should be observer-independent,
 - So that different people can agree on the reliability.



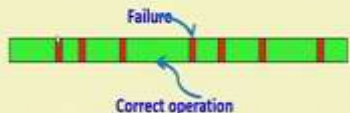
IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Of course one very basic issue with software is the different types of users use the system differently. And therefore, we cannot give different ratings to the same software. We need to give one reliability rating and that should be observer independent and all users should agree on that rating.

(Refer Slide Time: 10:41)

Mean Time To Failure (MTTF)

- Average time between two successive failures:
 - Observed over a large number of failures.



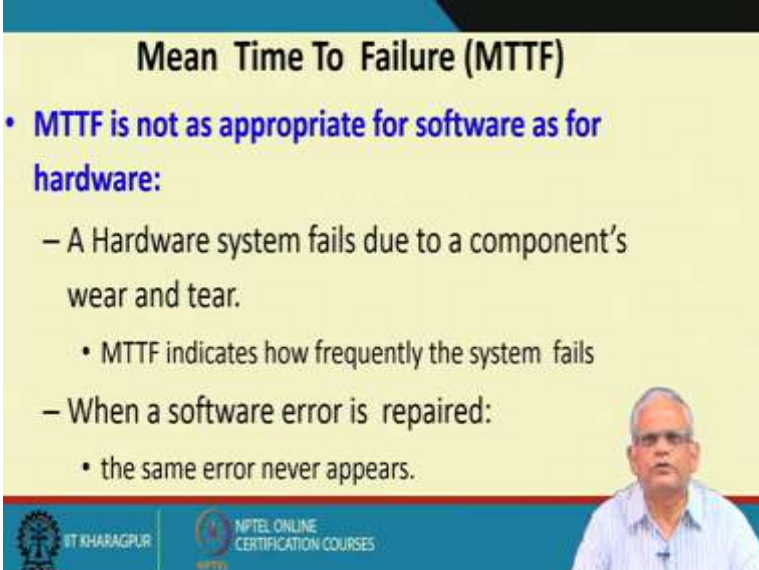
IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, let us see the reliability metrics for hardware, would they be useful for software. The mean time to failure is very popular hardware reliability metric, it is the average time between two successive failures or the inter-failure time. The mean inter-failure

time that is mean time to failure, here the software sorry the hardware this is a hardware metric we are trying to discuss whether you can use it for software. For a hardware system it is put to use for a long time and the failures are observed over this time and the average time to failure is computed from this.

For example, let us say this is the time of uses, there are correct operations after quite some time there is a failure. For a car may be the wheel has worn out and then again correct operation and then there is a failure, may be the battery has run out; replace the battery. And, then again there is a correct operation and then there is a failure may be the wire has got heated and shorted. So, replace the wire and again correct operation. The time between one failure to another failure is the inter-failure time and we can measure the inter-failure time for different failures and then compute the mean of that, that we will call as the mean time to failure.

(Refer Slide Time: 12:54)



Mean Time To Failure (MTTF)

- **MTTF is not as appropriate for software as for hardware:**
 - A Hardware system fails due to a component's wear and tear.
 - MTTF indicates how frequently the system fails
 - When a software error is repaired:
 - the same error never appears.

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES


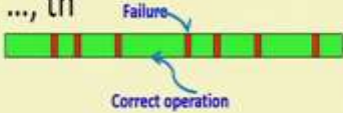
But, mean time to failure would not be very appropriate for software as per hardware because each time there is a failure, the bug is fixed and the rate of failure changes. For hardware it is a very reliable metric because the reliability is maintained and its meaningful to observe it for a long period and measure its reliability. But, for software reliability keeps changing, it improves typically, but then there can be glitches or the software may degrade due to a bug fix, the reliability may degrade and therefore, it either

increases or decreases. So, the mean time to failure where we observe the inter-failure times not be a really very appropriate metric.

(Refer Slide Time: 14:03)

Mean Time To Failure (MTTF) Measurement

- Record failure data for n failures:
 - let these be t_1, t_2, \dots, t_n
 - calculate $(t_{i+1} - t_i)$
 - the average value is MTTF

$$\frac{\sum(t_{i+1} - t_i)}{(n-1)}$$



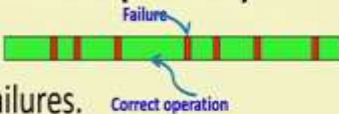
IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

For a hardware system typically the failure times are recorded, let us say the failures occur at t_1, t_2, t_n and then the inter-failure times are computed. And, then they are summed the total number of inter-failure times and divided by the total number of failures. So, computing the MTTF is straight forward. $MTTF = \frac{\sum(t_{i+1} - t_i)}{(n-1)}$

(Refer Slide Time: 14:46)

Rate of occurrence of failure (ROCOF)

- ROCOF measures:
 - Frequency of occurrence failures.
 - Observe the behavior of a software product while in operation:
 - over a specified time interval
 - calculate the total number of failures during the interval.



IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

But, what about rate of occurrence of failure, this is another metric. Here the frequency of occurrence of failure is obtained. We observe the time period, long enough time period and find how many failures have occurred and then from that we find the number of failures per unit time. So, that is called as the rate of occurrence of failure. We just count how many failures and then the total time of uses and then find the number of failures per unit time or the rate of occurrence of failure.

(Refer Slide Time: 15:31)

Mean Time to Repair (MTTR)

- Once failure occurs:
 - Some time taken to fix faults
- MTTR:
 - Measures average time it takes to fix faults.

The diagram shows a horizontal green bar representing a system's operation. The bar is divided into segments by vertical red lines. An arrow labeled 'Failure' points to one of the red lines. Another arrow labeled 'Correct operation' points to the green segments between the red lines. The slide also features the logos of IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, and a small video inset of a man in the bottom right corner.

But, another issue is that once a failure occurs, it takes some time to fix the bug. There is a repair time because each time there is a failure, there is a repair time. For a hardware may be need to identify which component has failed and then replace the component. And for software debug, find out the bug, correct the code; but in both cases hardware and software there is a time to repair which is the time to fix the bug. So, each time there is a failure some time is needed to fix it, that we call as the mean time to repair.

(Refer Slide Time: 16:33)

Mean Time Between Failures (MTBF)

- We can combine MTTF and MTTR:
 - to get an availability metric:
 - **MTBF=MTTF+MTTR**
- MTBF of 100 hours would indicate:
 - Once a failure occurs, the next failure is expected after 100 hours of clock time (not running time).

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

If we consider mean time to repair, in the previous case we just considered mean time to repair is 0. But, if we really consider the time to repair then the mean time between failure is the mean time to failure plus the mean time to repair. This is the time a system is not used, mean time to repair and mean time to failure is the time the system is used. And, the mean time between the failure is once the failure occurs when is the next failure expected that is MTBF is MTTF plus MTTR. $MTBF=MTTF+MTTR$

If we say that MTBF of 100 hours; that means, that once a failure occurs the next failure will occur after 100 hours and this includes the downtime of the software. Once a failure occurs the next failure is expected after 100 hours of clock time, that is including the downtime and it is not the run time of this software.

(Refer Slide Time: 17:57)

Probability of Failure on Demand (POFOD)

- Unlike other metrics
 - This metric does not explicitly involve time.
- Measures the likelihood of the system failing when a service request is made:
 - POFOD of 0.001 means:
 - 1 out of 1000 service requests may result in a failure.

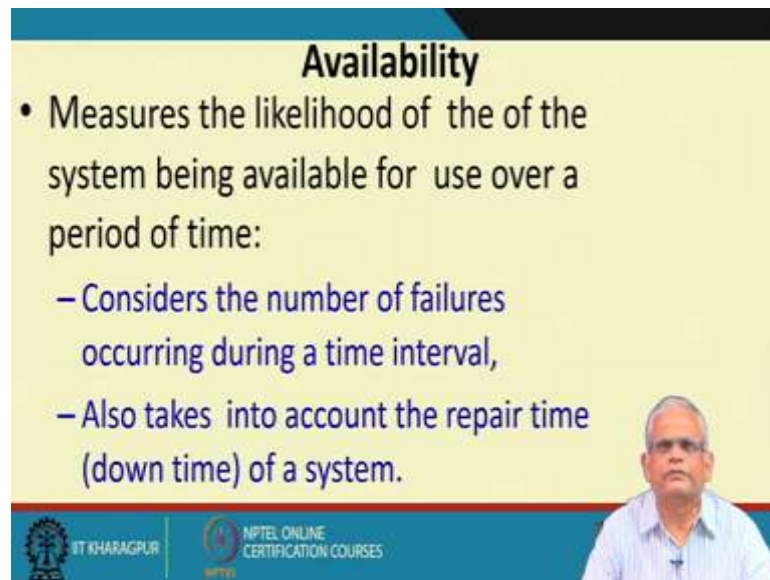
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

There is another metric, the probability of failure on demand; unlike the MTTF and rate of occurrence of failure, the probability of failure and demand does not involve a time. We do not measure it over a interval. Here we just keep on executing the software with different functionalities and found out, if we invoke the functionality with different data test input for 1000 times. How many times did the software fail?

If it is the probability of failure and demand is 0.001 that is on; that means, that if we executed the functionalities of the software 1000 times, then only one failure was observed. This is a more meaningful metric for software, because unlike a hardware which is continuously used, a software does not run unless a function is invoked. If the user invokes let us say each function, it runs for a fraction of a second or something and waits for the next input. So, the system is not running continuously unlike a hardware, let us say car is running for 10 hours, it is continuously the hardware is being used.

Whereas, here the software each time the user invokes a functionality, the software runs for a small time and then keeps waiting for the next input. And therefore, observing the software reliability over a period of time is less meaningful because, the question comes is that how frequently were the functions execute. Because, the software is not really running all the time, much of the time it was waiting for the user input. So, the probability of failure on demand is more appropriate metric for software.

(Refer Slide Time: 20:35)



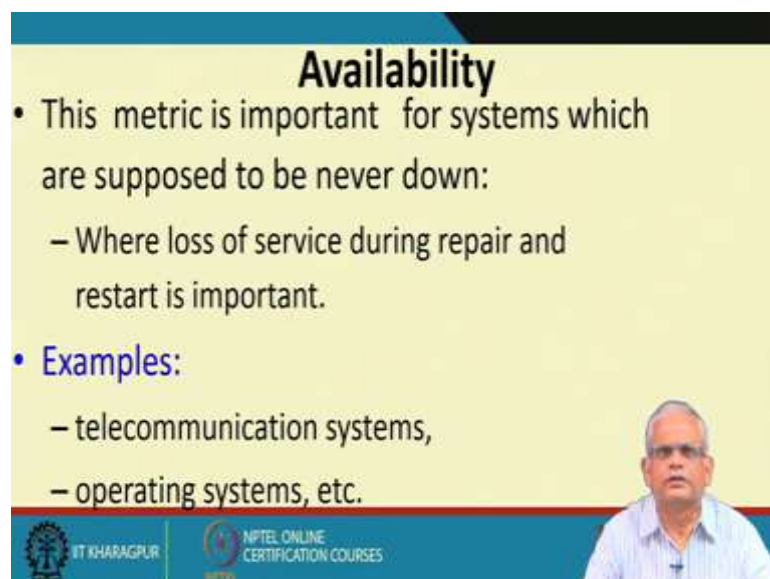
Availability

- Measures the likelihood of the of the system being available for use over a period of time:
 - Considers the number of failures occurring during a time interval,
 - Also takes into account the repair time (down time) of a system.

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

We can also define another metric called as availability which is basically characterizes the likelihood of the system being available for use to the user over a period of time. And of course, it will consider the failure time and the time to repair which we call as the downtime. And, based on this we can give a availability number that is a user once trying to use the software how likely that the software is available for use. Because, it may be undergoing, it might have failed, it might be undergoing debugging and so on.

(Refer Slide Time: 21:32)



Availability

- This metric is important for systems which are supposed to be never down:
 - Where loss of service during repair and restart is important.
- **Examples:**
 - telecommunication systems,
 - operating systems, etc.

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

The availability metric is important for systems which are not suppose to be down, they run continuously. For example, an operating system or a telecommunication software.

(Refer Slide Time: 21:48)

Availability

- Operational availability
$$A_o = \frac{MTBF}{MTBM + MDT}$$

MTBM = mean time between maintenance
MDT = mean down time
- Inherent availability
$$A_o = \frac{MTBF}{MTBF + MTTR}$$

MTBF = mean time between failures
MTTR = mean time to repair

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

But, then we have two notions of availability: one is the operational availability and the other is inherent availability. In the operational availability we just consider the mean time between maintenance and the mean down time, whereas, the inherent availability is the mean time between failure and the mean time to repair.

$$\text{Operational availability} = \frac{MTBF}{MTBM + MDT}$$

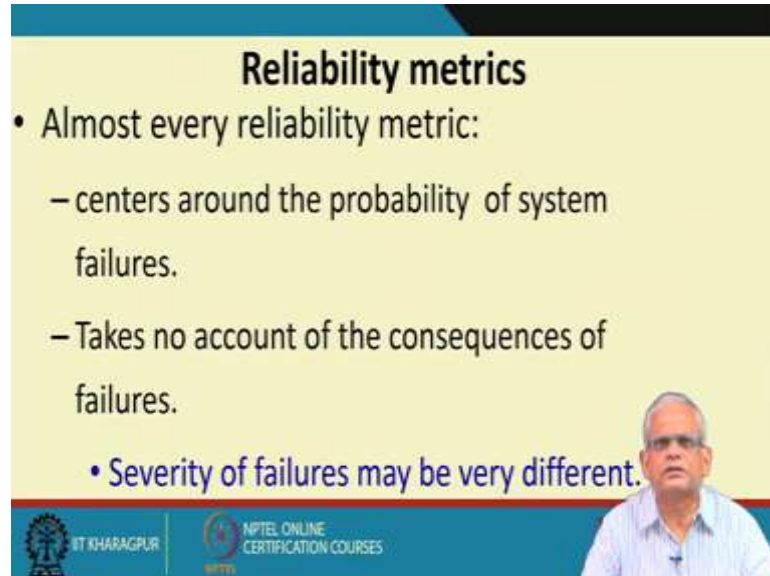
$$\text{Inherent availability} = \frac{MTBF}{MTBF + MTTR}$$

So, here we just consider the time between failure and the repair time, whereas here we consider the mean downtime. The downtime is different from the meantime to repair, in the sense that this is the time to repair.

But, then there may be preventive maintenance, there may be initialization study and so on why the system is down. So, the mean downtime is a more general measure of the downtime which not only the repair, but also the system may be down due to various reason including preventive maintenance, routine initialization etcetera. So, the operational availability is the actual availability to the user whereas, the inherent

availability is a ideal notion of availability; where we do not have preventive maintenance etcetera. The only downtime is due to the repair time.

(Refer Slide Time: 23:38)



Reliability metrics

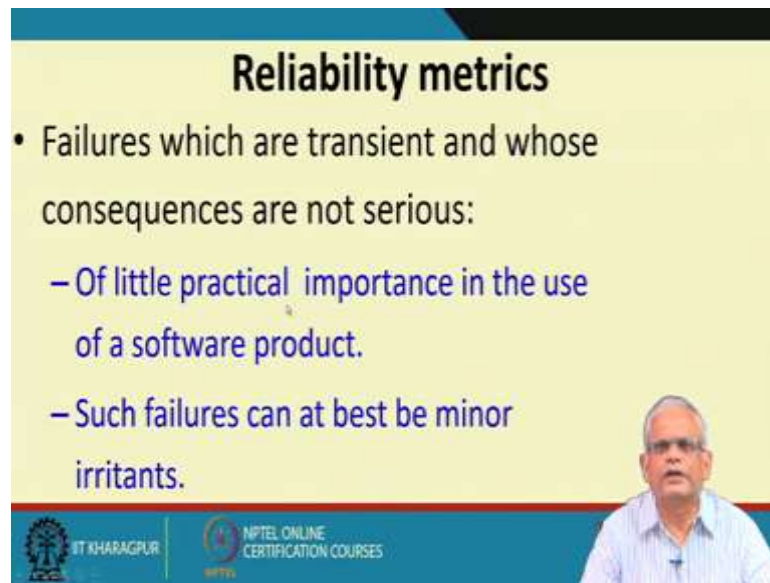
- Almost every reliability metric:
 - centers around the probability of system failures.
 - Takes no account of the consequences of failures.
- Severity of failures may be very different.

The slide includes a video inset of a man speaking in the bottom right corner. At the bottom, there are logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES.

But, if we just look back into all the metrics that we discussed, all the metrics are centered around the probability of the system failure. And, they do not take any account of the consequence of failure, but in reality some failures are very severe, they just the system hangs. We need to reboot, work is lost and so on. Whereas another function, another failure may be that the current system, the current function invocation did not work, but the system did not hang; just need to try different function or the same function the second attempt it may work and so on.

So, the severity of different failures may be different. If we just consider all failures occurring over a period of uses that may not be very proper. We cannot compute all types of failure and then come up with a reliability metric, because some may be very insignificant failures. And, if we only count the severe failures like cross type of failure that also will not be proper. So, how do we go about handling this problem? The different failures have different severities.

(Refer Slide Time: 25:10)



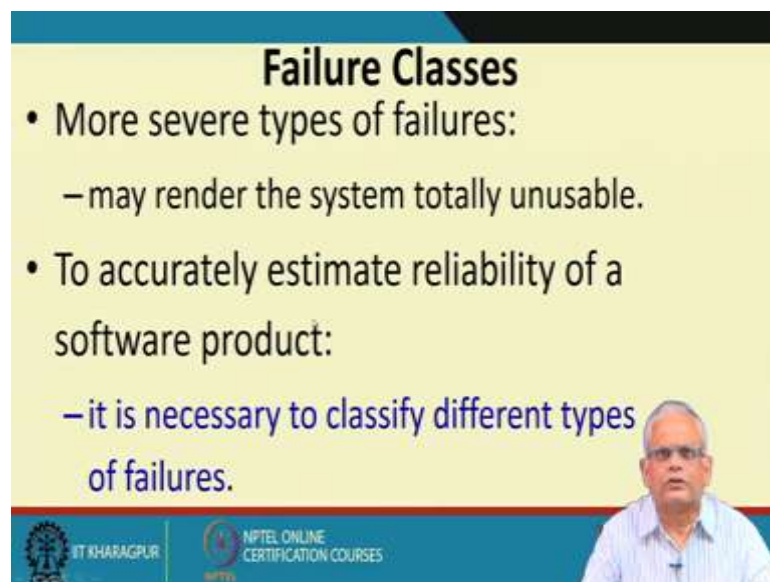
Reliability metrics

- Failures which are transient and whose consequences are not serious:
 - Of little practical importance in the use of a software product.
 - Such failures can at best be minor irritants.

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Some failures are transient and consequences are not serious and they are of little practical importance. At best they may be just minor irritants may be the mouse did not work, but next time the mouse worked and so on.

(Refer Slide Time: 25:41)



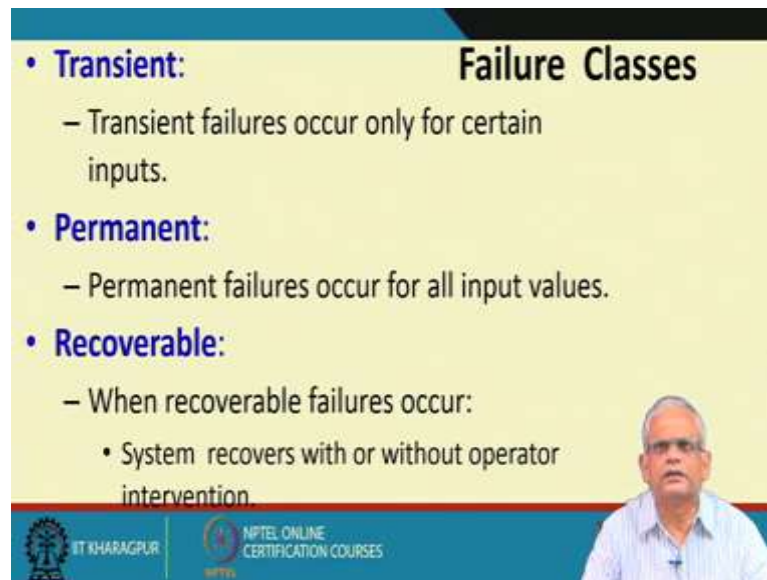
Failure Classes

- More severe types of failures:
 - may render the system totally unusable.
- To accurately estimate reliability of a software product:
 - it is necessary to classify different types of failures.

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

How do we handle this problem? We handle this problem by defining a failure class, different classes of failure.

(Refer Slide Time: 25:53)



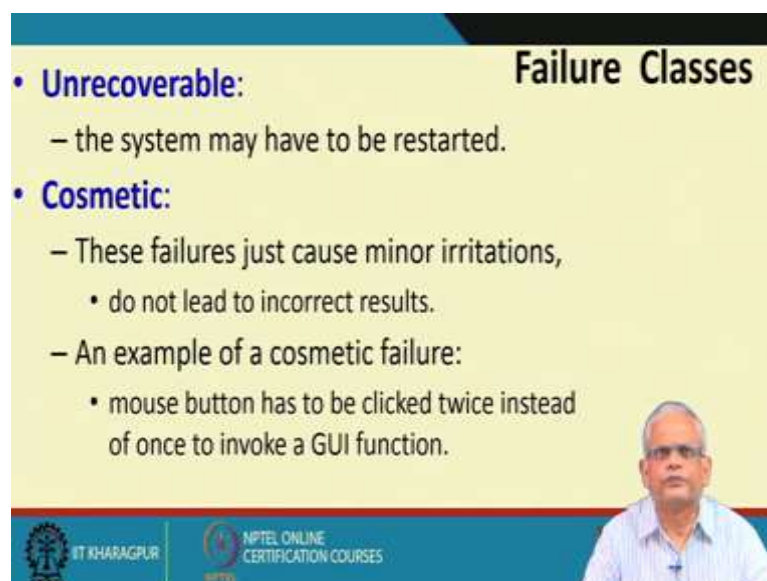
Failure Classes

- **Transient:**
 - Transient failures occur only for certain inputs.
- **Permanent:**
 - Permanent failures occur for all input values.
- **Recoverable:**
 - When recoverable failures occur:
 - System recovers with or without operator intervention.

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

We classify the different types of failure. Transient: this class of failure they occur only certain inputs, permanent failure they occur for all input values, some function they fail for any input value. But, let us say only fails per one input value when we give only 100 it fails, for all other it work satisfactory; then we call it as transient. Permanent it fails for all input values, recoverable that the system did not hang. We could recover either the user himself or with operator could recover and again without rebooting we could use the system.

(Refer Slide Time: 26:45)



Failure Classes

- **Unrecoverable:**
 - the system may have to be restarted.
- **Cosmetic:**
 - These failures just cause minor irritations,
 - do not lead to incorrect results.
 - An example of a cosmetic failure:
 - mouse button has to be clicked twice instead of once to invoke a GUI function.

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Unrecoverable the system might have hanged, we might have to restart the system those are unrecoverable. And, the cosmetic are the ones which are minor irritants, they do not really cause incorrect results or something, but just minor irritant may be the user need to sometimes press the mouse button twice. So, we can measure the reliability for each class of these failures and that will give a better idea of the reliability.

Maybe the reliability by considering the unrecoverable and the permanent of time type of failure or may be ignoring the cosmetic failure we can measure the reliability and so on. So, far we have looked at the reliability metrics and some very basic issues and then in the next lecture we will discuss about how to go about measuring the reliability of a software. We will stop at this point.

Thank you.