

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

Lecture – 59
Software Reliability – III

Welcome to this lecture. In the last lecture we had discussed about some issues in Software Reliability measurement.

(Refer Slide Time: 00:29)



In particular we discussed why software reliability estimation is a much harder problem compared to reliability estimation of other systems. We looked at the specific characteristics of software which make the reliability estimation difficult, we also discussed about few metrics which can be used to estimate the software reliability.

This lecture let us first look at software reliability estimation techniques. We will basically discussed two main techniques for software reliability estimation; one is reliability growth modelling and the other is statistical testing let us get started with that.

(Refer Slide Time: 01:31)

The slide features a yellow title box at the top with the text "How to Estimate The Reliability of a Software Application?". Below this, a list of three bullet points is displayed in blue text: "• Reliability Growth Modelling", "• Statistical testing", and "• Indirect means". At the bottom of the slide, there is a small inset video of a man with glasses and a blue shirt. The footer of the slide includes the logos for "IIT KHARAGPUR" and "NPTEL ONLINE CERTIFICATION COURSES".

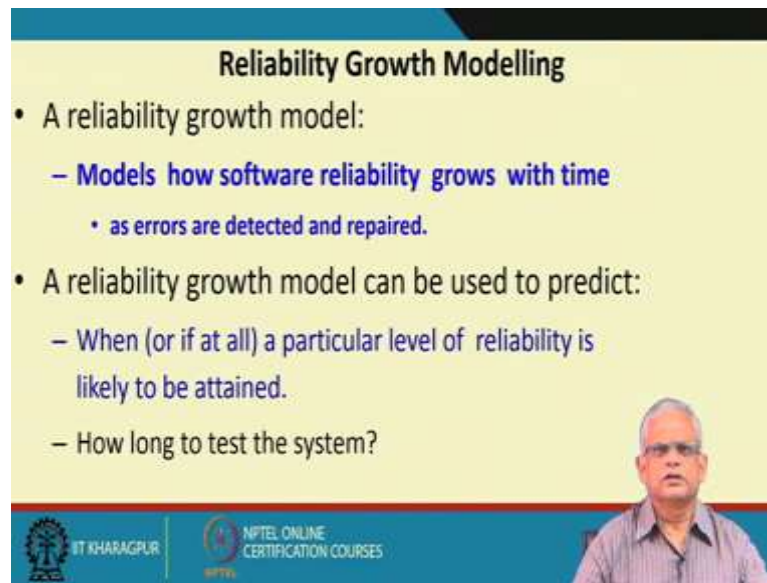
So, the problem that we are trying to address is that given a software application how do we go about estimating the reliability of the application? The application has been developed, installed and running. Our task is to estimate the reliability of the application, we need a quantitative value for the reliability of the application.

One of the options is reliability growth modelling. We just try to map the failure rate with a reliability growth modelling curve and based on that we can say that what is the reliability of the software after elapse of sometime at the movement so on. We will look at this technique reliability growth modeling as a way to estimate the reliability of a software application. We will look at statistical testing this is another way to estimate the reliability of the application.

The third technique is indirect ways of measuring the reliability here we do not really look at the failure rate and so on and measure the reliability, but based on the characteristics of the software. For example, various metrics of the software its complexity matrix number of calls per use cases per use case and so on.

So, based on a set of metrics that are collected for the application, we can predict the reliability. We will not look at the indirect means we will look only at the reliability growth modelling and statistical testing.

(Refer Slide Time: 03:55)



Reliability Growth Modelling

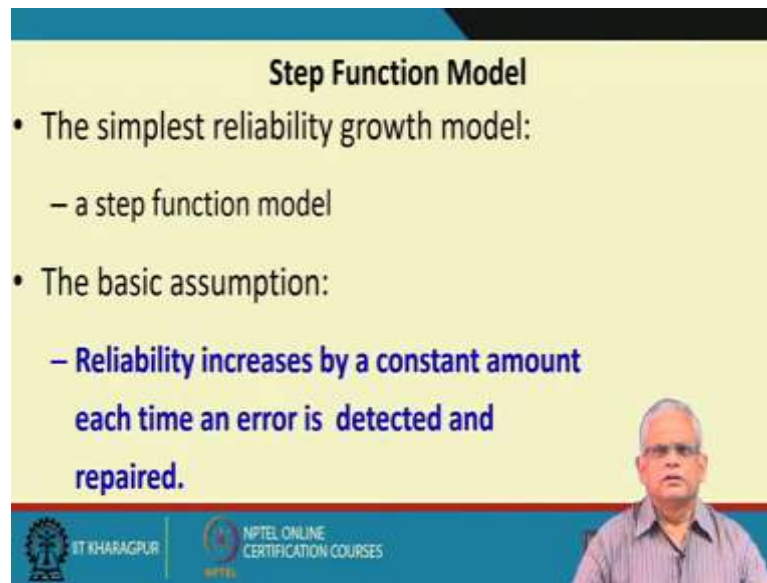
- A reliability growth model:
 - Models how software reliability grows with time
 - as errors are detected and repaired.
- A reliability growth model can be used to predict:
 - When (or if at all) a particular level of reliability is likely to be attained.
 - How long to test the system?

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Let us look at reliability growth modeling to start with. A reliability growth model is a graphical model of the reliability growth of the software with time. As different errors are detected and repaired the reliability of the software changes and we model over time how the reliability would grow.

The use of such a reliability growth model is that we can say that what will be the reliability of the software after some time. We can also use this model to predict that given a quantitative requirement of the reliability of the software how long will the testing need to go on. Because as testing goes on, more and more bugs are detected and fixed given that a certain level of reliability is given for the application. We can predict how long how much longer to test is it 2 weeks, a month, 2 month this is another use of a reliability growth model.

(Refer Slide Time: 05:31)



Step Function Model

- The simplest reliability growth model:
 - a step function model
- The basic assumption:
 - **Reliability increases by a constant amount each time an error is detected and repaired.**

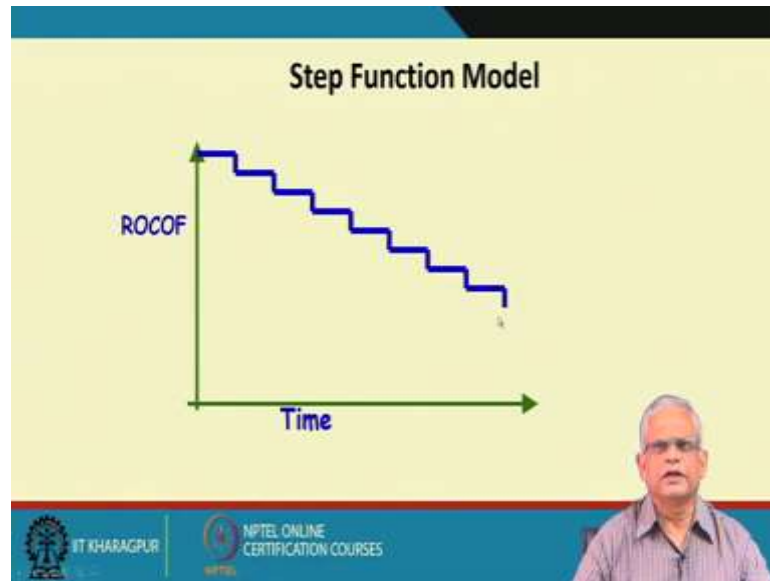
IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

The simplest growth model is the step function model its quite old model very simple, but then it is also intuitive and forms a background for other more sophisticated reliability growth models. Here the basic assumption is that each time a failure occurs the corresponding error is detected and fixed and the number of errors in the software decreases therefore, the reliability increases.

The simple assumptions made by this model is that all error fixes are ideal and error fix permanently fixes the error there are no side effects of that no other bugs are introduced by that and also each time an error is fixed, the reliability increases by a constant amount. Very simplistic model we know already based on our intuitive knowledge of the software reliability that reliability growth on an error fix is not constant.

Initially the bugs that are there on the core of the software get exposed they cause failure because those are the ones which cause frequent failures and they are the ones to get detected first. And, fixing the bugs in the core part increases the reliability by a rather larger amount compared to fixing a bug that is there in a remote part of the software. So, the assumptions are too simple, but then this can be used to predict reliability with some approximation.

(Refer Slide Time: 07:51)



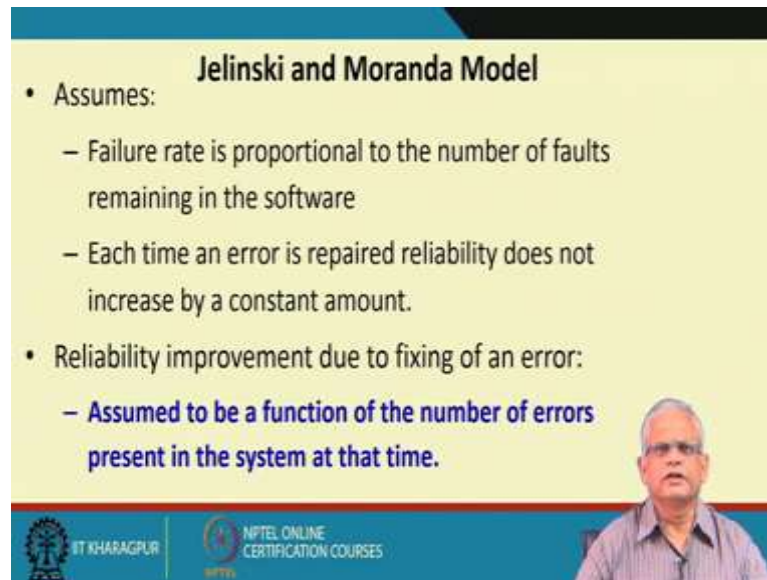
The graphical representation of the model is this step function, the rate of occurrence of failure on the y axis and time and the x axis and each time there is a failure, the rate of occurrence of failure comes down reliability improves. And, based on how frequently are errors discovered and so on we can fit onto this model and predict the reliability of a given application, this is the simplest model.

(Refer Slide Time: 08:39)

-
- The figure is a slide titled "Step Function Model". It contains a bulleted list of assumptions and unrealistic aspects. The slide also includes a small video inset of a man in the bottom right corner and logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES at the bottom.
- Assumes:
 - All errors contribute equally to reliability growth
 - Unrealistic:
 - We already know that different errors contribute differently to reliability growth.

Here one of the implicit assumption is that all errors contribute equally to the reliability growth we know from our basic understanding that different errors contribute differently to reliability growth. And therefore, the results of this model will not be very accurate.

(Refer Slide Time: 09:01)



Jelinski and Moranda Model

- Assumes:
 - Failure rate is proportional to the number of faults remaining in the software
 - Each time an error is repaired reliability does not increase by a constant amount.
- Reliability improvement due to fixing of an error:
 - Assumed to be a function of the number of errors present in the system at that time.

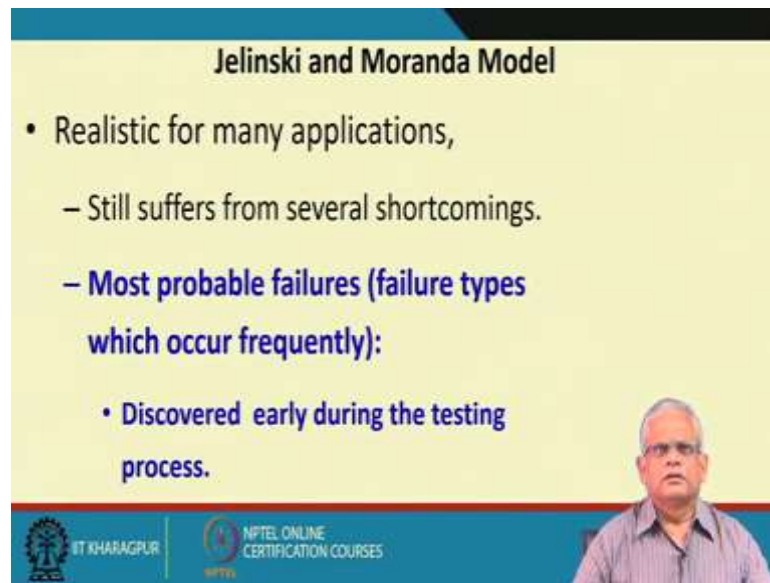
IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Another basic model is the Jelinski Moranda model again proposed long time back. Here the basic assumption is that the failure rate is proportional to the number of faults remaining in the software. Again implicit assumption being that each error is responsible to cause failure at the same rate.

So, the failure rate is proportional to the number of faults remaining in the software and each time an error is repaired reliability does not increase by a constant amount. We can see here since the failure rate is proportional to the number of faults remaining in the software. If n is the number of faults, the failure rate is the function of n , it is proportional. So, let us say $k n$ and the failure rate after bug has been detected and corrected is n minus 1, $k n$ minus 1.

We can see that the improvement in the reliability is not constant here and is a function of the number of bugs. Here initially the improvement in reliability will be low on a bug fix because out of n bugs we fixed one bug. So, basically proportional to 1 by n , the, but towards the end if there are two bugs and we fixed one of that the reliability growth will be higher this is the assumption here.

(Refer Slide Time: 11:27)



Jelinski and Moranda Model

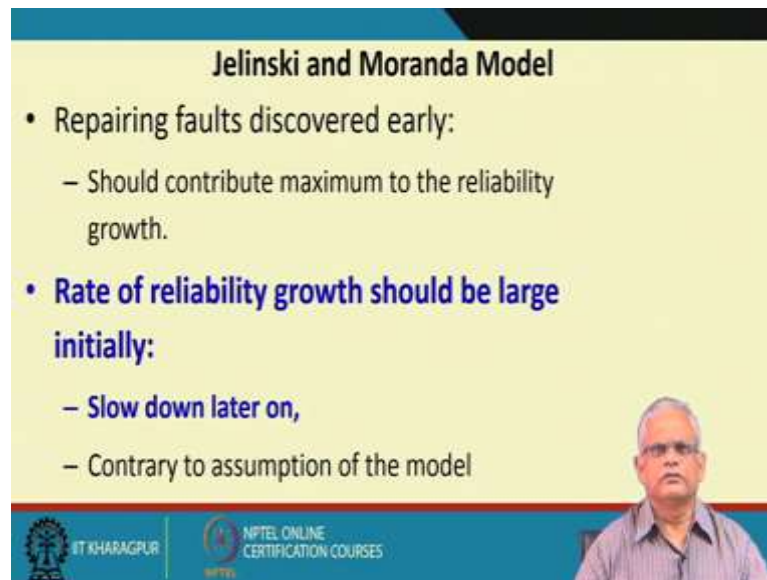
- Realistic for many applications,
 - Still suffers from several shortcomings.
 - **Most probable failures (failure types which occur frequently):**
 - **Discovered early during the testing process.**

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Its better than the step function model here each bug fix contributes differently to the reliability growth, but then still there are several short comings. As we are saying that intuitively we know that the bugs that are there in the core part get discovered early contribute to larger growth of reliability.

But then as time passes by the bugs that are there on the noncore part of the software get detected and these contribute to lower increase in reliability. And therefore, initially the reliability growth should be high and later part the reliability growth should be slow down, but here this model does not do that.

(Refer Slide Time: 12:49)



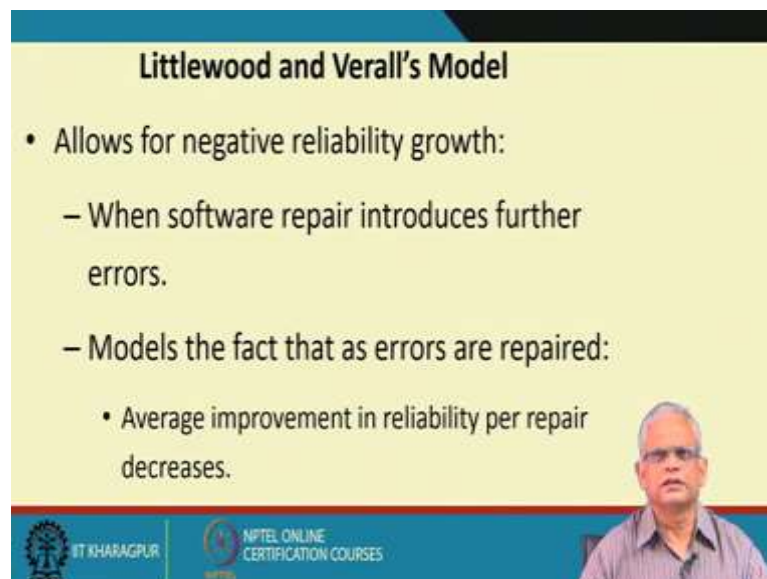
Jelinski and Moranda Model

- Repairing faults discovered early:
 - Should contribute maximum to the reliability growth.
- **Rate of reliability growth should be large initially:**
 - **Slow down later on,**
 - Contrary to assumption of the model

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Initial faults that are detected and corrected should contribute maximum to the reliability growth and the rate of reliability growth should be large initially and slowed down later on and this is contrary to the assumption of this model. And therefore, this model will not really give very accurate results.

(Refer Slide Time: 13:13)



Littlewood and Verall's Model

- Allows for negative reliability growth:
 - When software repair introduces further errors.
 - Models the fact that as errors are repaired:
 - Average improvement in reliability per repair decreases.

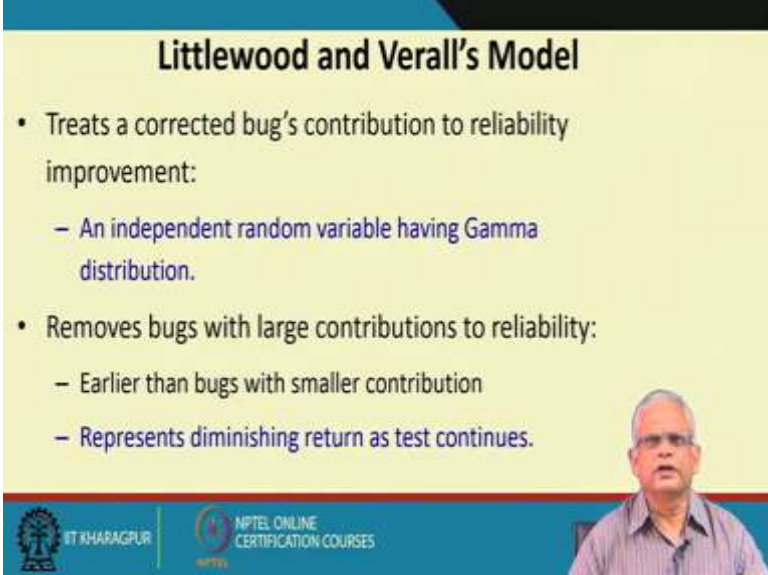
IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, let us look at the Littlewood and Verall's model, in both the models we have seen So far, the step function model and the Jelinski Moranda model both it was assumed that the bug fixes are ideal, that is when a bug is fixed error is reduced. But in reality on a

bug fix more errors may get introduced that bug may get fixed, but it may cause more error to get introduced. And therefore, the number of bugs may actually increase. And therefore, on a bug fix the reliability may actually decrease there is a negative reliability growth this really happens practically and this is taken care in the Littlewood and Verall's model.

Negative reliability growth may occur sometimes when the repair introduces further errors. And here as we can see that the growth in reliability is not constant and also the average improvement in reliability per repair decreases because sometimes there are negative reliability growths.

(Refer Slide Time: 14:53)



Littlewood and Verall's Model

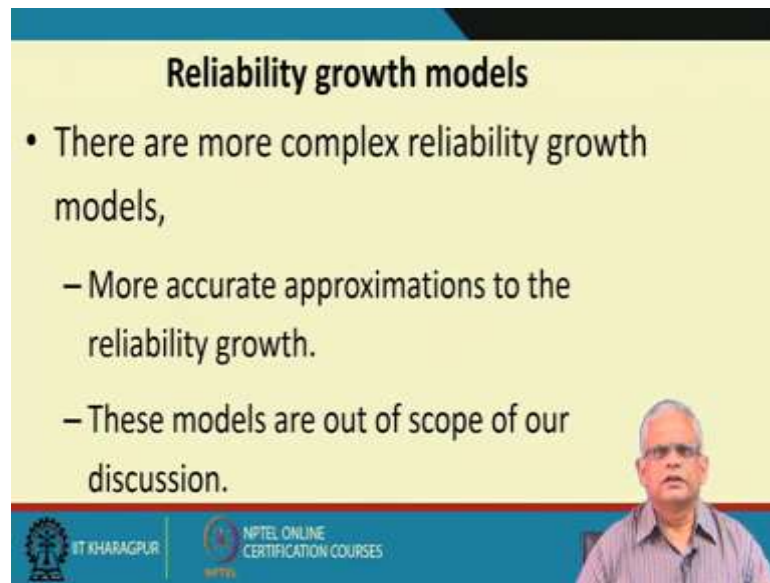
- Treats a corrected bug's contribution to reliability improvement:
 - An independent random variable having Gamma distribution.
- Removes bugs with large contributions to reliability:
 - Earlier than bugs with smaller contribution
 - Represents diminishing return as test continues.

The slide includes a video inset of a man in the bottom right corner and logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES at the bottom.

To do this the model treats a corrected bugs contribution to the reliability as a independent random variable having gamma distribution. And here the bugs with large contributions to the reliability are considered initially these the bugs that are initially detected and removed these contribute to higher reliability growth than the bugs that are detected later.

So, this matches with the intuitive understanding of the reliability growth. It represents diminishing return as the test continues and this model of course, matches with the intuitive idea of the reliability growth. And therefore, would give a more accurate result than both the step function model and the Jelinski Moranda model.

(Refer Slide Time: 16:01)



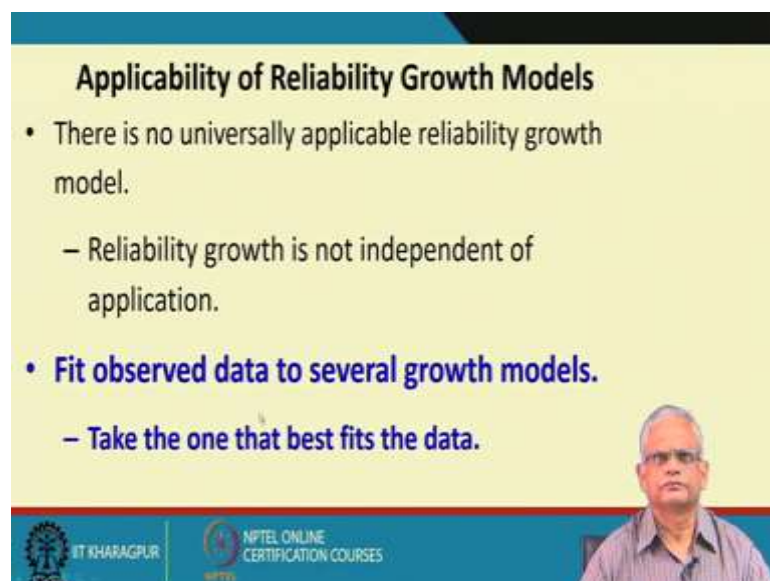
Reliability growth models

- There are more complex reliability growth models,
 - More accurate approximations to the reliability growth.
 - These models are out of scope of our discussion.

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

There are several dozens of reliability growth models that have been proposed based on these simple reliability models. These provide more accurate approximations of the reliability growth, but then ours being a project management course we are more interested in getting the basic ideas on the reliability growth the basic models rather than really discussing very sophisticated models and spending time on that. So, these more sophisticated models are out of scope of our discussion.

(Refer Slide Time: 16:45)



Applicability of Reliability Growth Models

- There is no universally applicable reliability growth model.
 - Reliability growth is not independent of application.
- **Fit observed data to several growth models.**
 - **Take the one that best fits the data.**

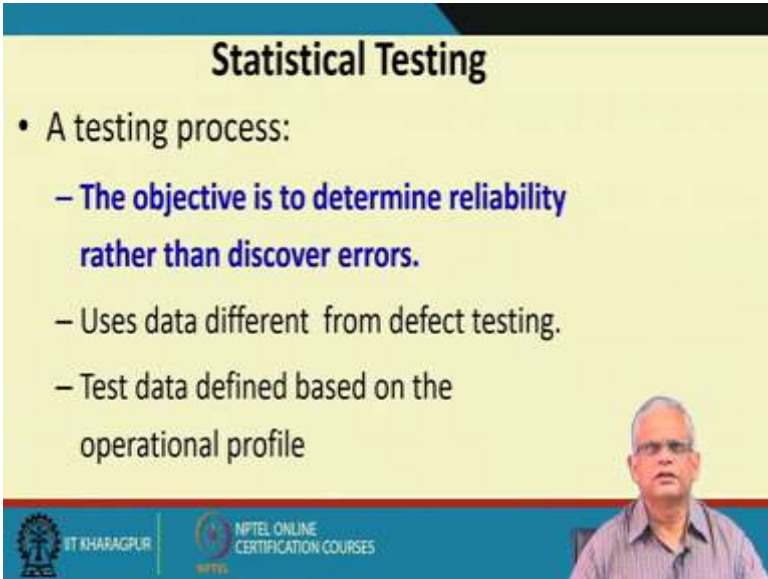
IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

But given that there are several dozens of models, each making some simplifying assumptions and some realistic assumptions and so on. How do we determine which model to use, is it that there is a best model which will be used for all applications that will give the best result? No not really because the reliability growth is dependent on the specific application that we are considering.

The bug distribution the number of functions supported the complexity of the software and so on. And therefore, we cannot straight away say that this model is suitable for this type of software, but what we can do is given the failure data over a time period a considerable time period we can try to plot it and match with these growth models.

The one it matches most accurately that will be the one to be used and the best fit model is taken and based on that we can say that after elapse of certain time what will be the reliability of the software.

(Refer Slide Time: 18:23)



Statistical Testing

- A testing process:
 - **The objective is to determine reliability rather than discover errors.**
 - Uses data different from defect testing.
 - Test data defined based on the operational profile

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

Now, let us look at another technique the statistical testing technique, this is a testing process no doubt, but then here the objective is not to detect bugs. Here the objective is to determine reliability and if some bugs get detected that is not the focus, but then that is good that some bugs get detected, but here the objective is to determine the reliability of the software. And naturally the test case design here is different than the defect testing. One thing we can say that the test data here these are designed based on the operational profile of the software.

Since the test data is defined based on the operational profile of the software this gives a numerical value for the reliability of the software which is agreeable to most users. We had said that the reliability of a software is observer dependent, different observers may have a different estimation of the reliability of the software. And we had said that it will be confusing if we give different reliability values for the same software for different classes of users we need to give only one value.

And the value that we can give is the average of all the users and that is captured here in the operational profile, this is the average rate at which the different functions of the software are invoked considering all the users.

(Refer Slide Time: 20:27)

How to define operational profile?

- **Divide the input data into a number of input classes:**
 - e.g. create, edit, print, file operations, etc.
- **Assign a probability value to each input class:**
 - Based on the probability that an input value from that class to be selected for a test case.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

We had discussed about the operational profile in one of our earlier lectures. Here we look at the different functionalities like a let us say word processing software may have the functionalities create document, edit document, print file operation etcetera. And then we find out the test data which correspond to create, edit print file operation etcetera. And we observe how the software gets used by different users.

Maybe we can write a small log collecting software which will keep on collecting the log of uses by various users of the software and from there we can find out what is the rate at which these different functions are invoked by different users. And based on this log we can assign a probability value to each input class, each input class basically belongs to some of the operation. And then we will have all the functionalities of the software and


probability of an input value getting selected from that class that will form the operational profile.

(Refer Slide Time: 22:05)

Steps involved in Statistical testing (Step-I)

- Determine the operational profile of the software:
 - This can be determined by analyzing the usage pattern.

Function	Probability
f1:f11	0.4
f12	0.1
f13	0.1
F2:f21	0.05
f22	0.15
F3:f31	0.15
f32	0.05



IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, here in statistical testing the first step is to determine the operational profile of the software, the user's pattern can be captured by a log and then we can analyze to assign probabilities to each of the functions supported by the software.


Typically we will have various functions supported by the software and also different scenarios for the functionalities and then we give a probability of the function being invoked. This forms the operational profile definition of the software.

(Refer Slide Time: 22:59)

Statistical testing: Step 2

- Manually select or automatically generate a set of test data:
 - Corresponding to the operational profile.

Function	Probability
F1-f11	0.4
f12	0.1
f13	0.1
F2-f21	0.05
f22	0.15
F3-f31	0.15
f32	0.05



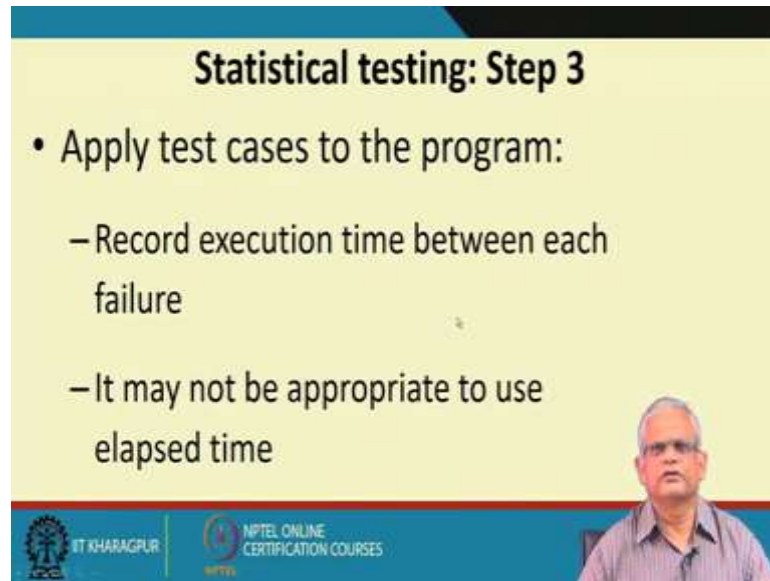
IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

The second step is to generate the test data; here the test data should correspond to the operational profile. What we mean is that if 40 percent 0.4 is the probability of the first scenario of F1 functionality, then in our test suite, we should have 40 percent of the test cases where F1 is invoked on the f11 scenario.

Similarly, 0.1 is the probability of the f12 scenario of functionality F1 and therefore, 10 percent of the test cases should correspond to the f12 scenario of the F1 functionality. As you can see here that if we analyze the test suite we should find a correspondence with the actual uses of the software. In the defect testing we do not do this we just anticipate where defects might be there and we design test cases, here we do not do that. Here based on the user's pattern we generate test data.

If f11 scenario of the F 1 functionality has a probability of 0.4 in our test suite design, we design 40 percent of the test cases with test values that correspond to the f11 scenario of the F1 functionality.

(Refer Slide Time: 24:49)



Statistical testing: Step 3

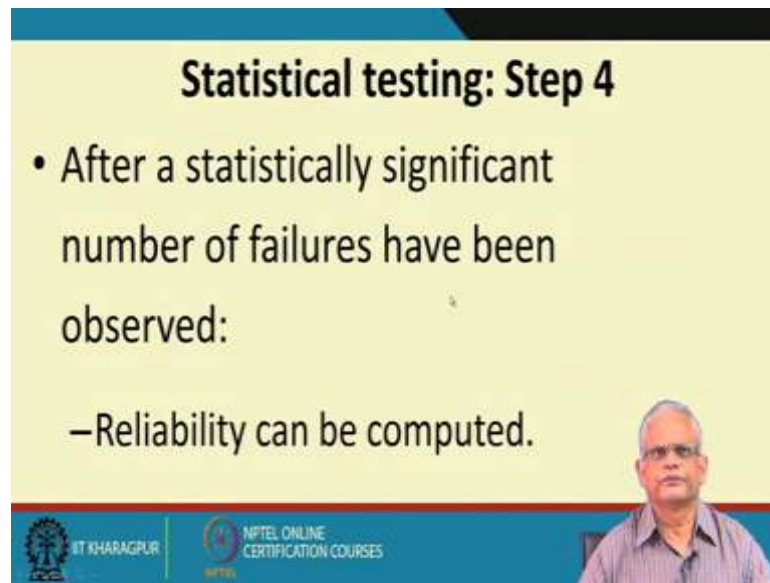
- Apply test cases to the program:
 - Record execution time between each failure
 - It may not be appropriate to use elapsed time

The slide features a video inset of a man with glasses speaking in the bottom right corner. The footer contains the logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES.

And in the step 3 actually carry out the execution of the software using the test cases and find out the failures. But then measuring the clock time may not be the right thing to do we should actually measure the execution time. As we are discussing earlier, that the software runs very fast. Each functionality it typically completes in milliseconds, but the human input may be let us say you can give a input in several seconds or minutes.

And therefore, most of the time the software idles on each input it runs for a very short time and then idles for the next input. Therefore we should have some instrumentation some means by which we collect the actual run time information of the software and that we plot on our failure data, we consider that as the time of failure not just the clock time. Using clock time we will result in a very inaccurate reliability estimation.

(Refer Slide Time: 26:31)



Statistical testing: Step 4

- After a statistically significant number of failures have been observed:
 - Reliability can be computed.

The slide features a yellow background with a blue header and footer. A small video inset of a man with glasses is visible in the bottom right corner of the slide area. The footer contains the logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES.

And finally, after statistically significant number of failures have been observed the reliability can be computed that is how many functionalities were invoked and how many failures were observed and so on.

Here the result is likely to be agreeable to most of the users because the test data have been defined based on the operational profile. We are at the end of this lecture we will stop here and continue in the next lecture.

Thank you.