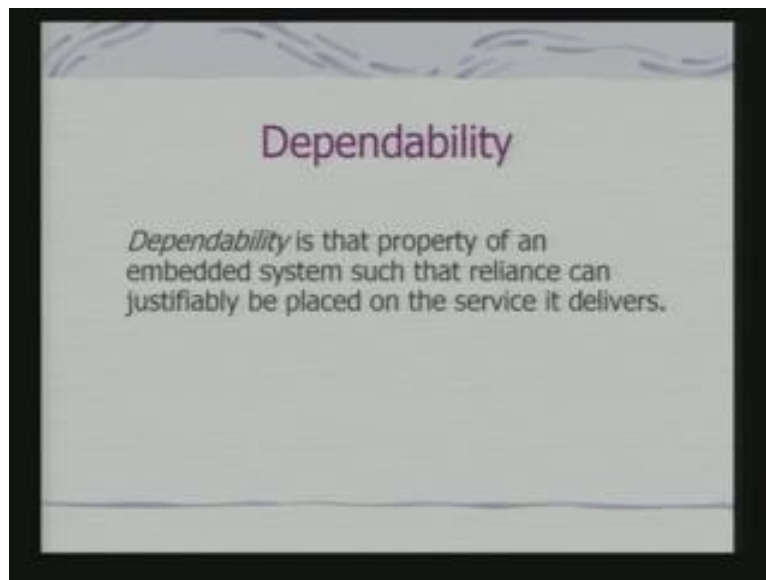


Embedded Systems
Dr. Santanu Chaudhury
Department of Electrical Engineering
Indian Institute of Technology, Delhi

Lecture - 36
Building Dependable Embedded Systems

Today, we shall consider another aspect of embedded system design that is the dependability aspect. Dependability is, that property of an embedded system such that reliance can justifiably be placed on the service it delivers.

(Refer Slide Time: 01:11)

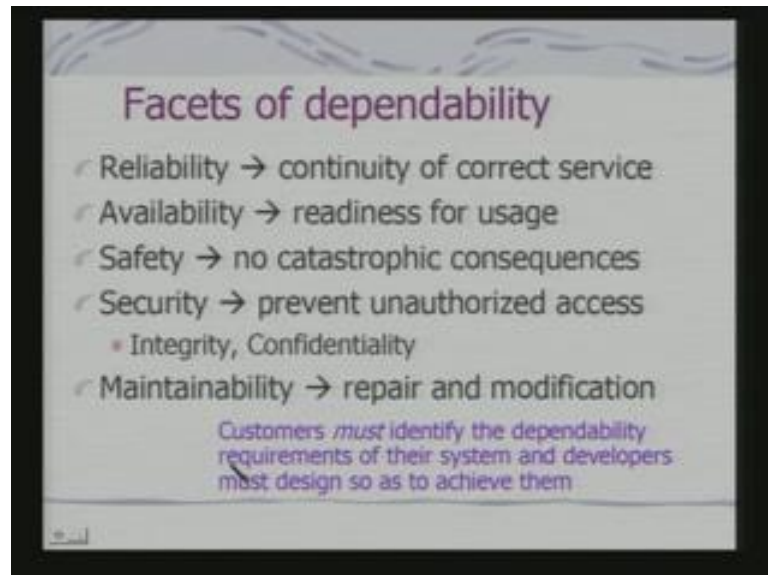


So, what does that really mean, it means; that when you are deploying embedded systems for various practical applications. We got to have some kind of a confidence about the functioning of the system; we might have designed a very sophisticated system. But, the system fails regularly then that system is violating the dependability characteristics. In fact, today when these kind of embedded systems are computer as components is being used in a variety of scenario.

This computers are also getting connected via network under these circumstances dependability is turning out to be a critical issue. Because we need to work in a fail safe fashion as well as, we need to protect our system against. So, these aspects of the design with relation to embedded systems is what we shall explore today. We obviously, shall

not have time to go in to details of the techniques, we shall basically have an exposure to different aspects of dependable design. There are various facets of dependability.

(Refer Slide Time: 02:52)



In fact the points that, I have been indicated here: reliability, availability, safety, security, maintainability all of these contribute towards dependability of an embedded system. A system is reliable if it continuous to provide correct service. Availability is the system is ready to provide you with the service when you are asking for it. If you consider a kind of a network attack an intrusion onto a system, such that the system gets clocked and when you are asking for a service, the server is not being able to provide the service then what we have is actually a service attack.

That is an example where, availability of a system can go down because of a loop goal in the system asserts. Safety refers to another aspect of dependability because, you would like the embedded system to work such that even if fails. There are no catastrophic consequences even if there are wrong inputs there are no catastrophic consequences. Related to this is security in fact the example of availability that, I have given you is the reason for the system becoming non available was a security. So, security is related to methods and techniques being built into the embedded systems to prevent unauthorized access.

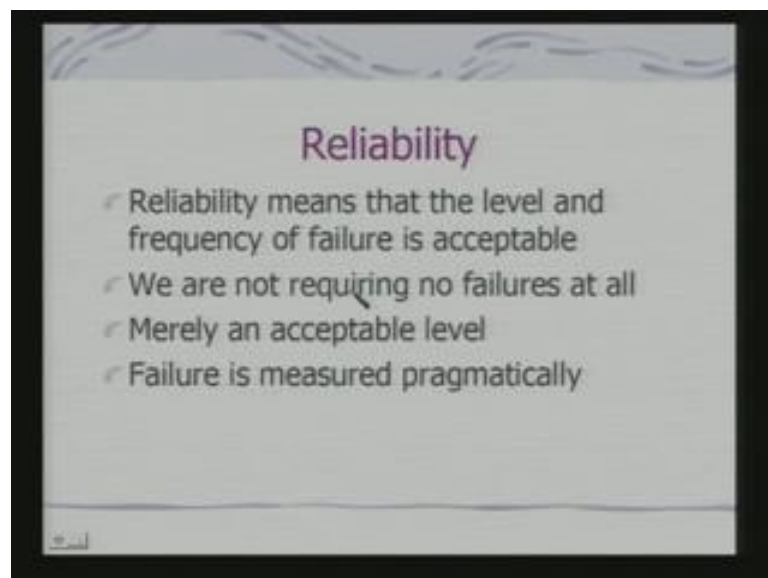
This is related to what is called integrity of the system as well as confidentiality of the data that is being stored as well as data being transmitted from the system. The next issue

is that of maintainability whether, the system can be repaired whether the software or the hardware can be upgraded on demand that also is an important issue for the purpose of dependability. Because if the system face and if cannot be repaired easily then the availability of the system goes down.

Because, a system may fail and if it has a mechanism to take care of the fault and restart its function within a short interval that the system can remain available at a regular basis. So, availability of the system is much better when the system is maintainable. So, all this facets together defines dependability of a system, but, depending on applications these factors can play different role. So, that is why we say the customers must identify the dependability requirements of the system and developers must design.

So as to achieve them. So, because you could realize that the movement, I am going for dependable design, I might need to do something extra, I might need to put extra hardware, I might need to modify my software. All this extra list to cost. So, whether the customer would bother for that additional dependability feature or not is a key issue in designing for dependability and depending on the appliance on the requirements. Different factors can be associated with different significance.

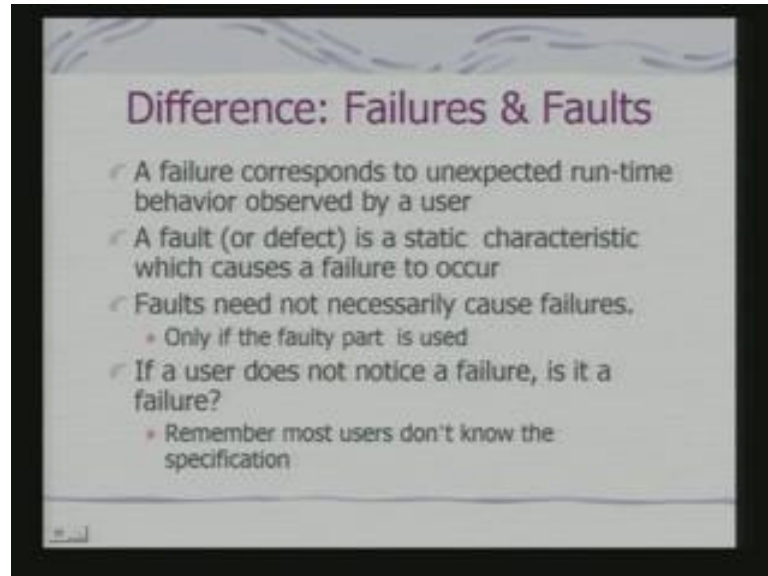
(Refer Slide Time: 06:37)



What is reliability? Reliability means: that the level and frequency of failure is acceptable; that means, we are assuming that there may be failures. So, we are not requiring that there is no failure at all. Because, any practical system cannot work really

without failures what we had asking for is an acceptable level of the failures and the failures are needed to be measured in a pragmatic fashion.

(Refer Slide Time: 07:12)



So, you should now understand therefore, difference between failures and faults a failure corresponds to unexpected run time behavior observed by a user. A fault on the other hand is the static characteristic, which causes a failure to occur. But it is also true faults need not necessarily cause failures; faults will cause failures only if the faulty part is used, this is true for software as well as for the hardware. Because, there may be a bug in a software, but, that part of the software is being really used. If that part is being really used then that failure would really occur.

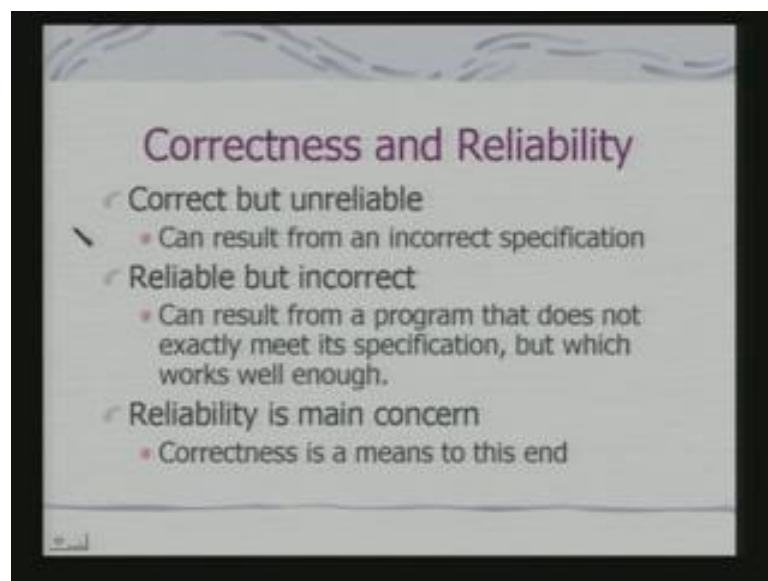
So, although that bug of the faults remains, in the software the system still can be reliable because, that part of the software is being really used, because you have to accept 1 reality that; we cannot eliminate failures. Failures can happen, because of certain transient phenomena in fact one example. We have talked about earlier that there may be in an SOC or on those parts may be really failure prone. So over a number of operations there can be faults transient, faults coming in failure of the components.

So, that failure is not really, because of a fault that failure has coming because some component has actually failed ok. Now, the question is if I really used a reliable component; that means, the probability of failure of that component is very low then, my system would be reliable same thing is true for the software. There may be bugs; bugs

cannot appear suddenly in a transient fashion. Bugs will be there in the software, because bugs were there when the software is design itself. And the testing process cannot always on all possible bugs.

If bugs remain in a really used part of the software then, that bug or the fault can lead to failure only when that part of the software is being used. And the other question is user may not always notice a failure, because user does not know the specification of the system. There may be a failure because of non compliance with, the specification of the system, but, if grossly the functional requirements have been met the user may not be able to detect the failure, because user does not know the specifications; see if the user does not detect the failure, in that case those failures can go completely on noticed. Let us look at the relationship, between correctness and reliability. A system can be correct but, unreliable.

(Refer Slide Time: 10:10)



Now, this can result from incorrect specification; how can I come, because when I am providing the specification; my specification can be incomplete in the sense that it provides: the functional requirements for the normal operational scenario. There may be some exceptions and those exceptions might not have been covered by the specification. Therefore when I write my software or design my hardware I have taken care of the basic requirement and the system is correct with respect to the basic requirement.

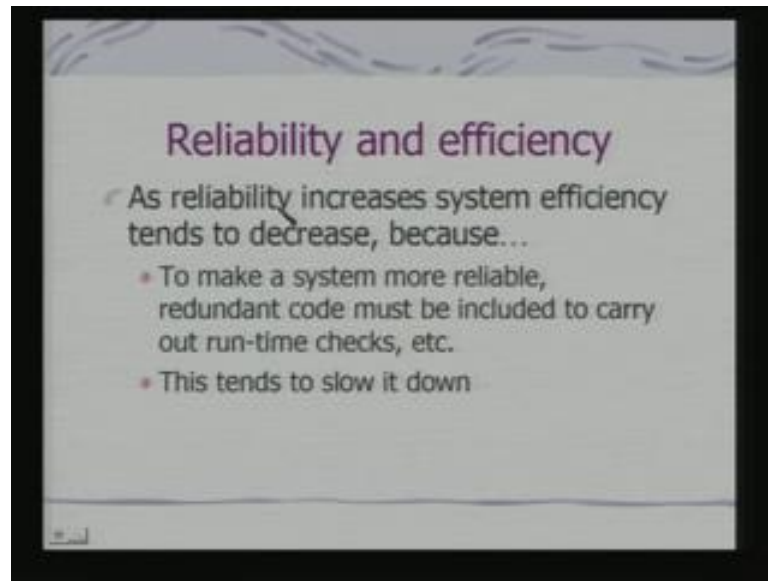
But the system has not taken care of some of the exceptional situations and when the exceptional situations raise the system fail. So, the system actually satisfies the specification, the system is correct, but, since the system fails under exceptional conditions and those exceptional conditions are not rare that is why the system can be unreliable. The system can be reliable, but, incorrect. So, it is an example of software can result from a program that does not exactly meet its specification, but, which works well enough; that means, it does not really have the bug.

But, it does not satisfy the functional requirements completely. So, in that case the system can be reliable, but, not correct. So, in fact, what we are looking for is a correct as well as or reliable system. So, reliability in a way is the main concern and correctness is the means; to this end why because the moment I can provide a proper specification and if I write my software or if I design my hardware meeting completely the specification and getting the correct system.

If I have got the correct system, if my specification is complete enough then I have the ability to take care of exceptional situations, which would increase the reliability of the system. But, we have to keep in mind that whatever, exhaustive testing I do there is the limitations in time; I cannot possibly take care of all possible combinations to provide that as a test data to the system to verify whether, it works properly under all possible inputs. So, there may be bugs they are already in the system ok.

So, my basic point is that I may design a system meeting the specifications. If my specification is complete, it will take care of most of the exceptional conditions as well as basic functional requirements, but, there may be failures. Because of some faults which go undetected there may be failures because of some transient phenomena leaving to failure of components. And we have to look at measures to increase the reliability of the system by taking care of these possible failure scenarios. Then the question comes of reliability and efficiency.

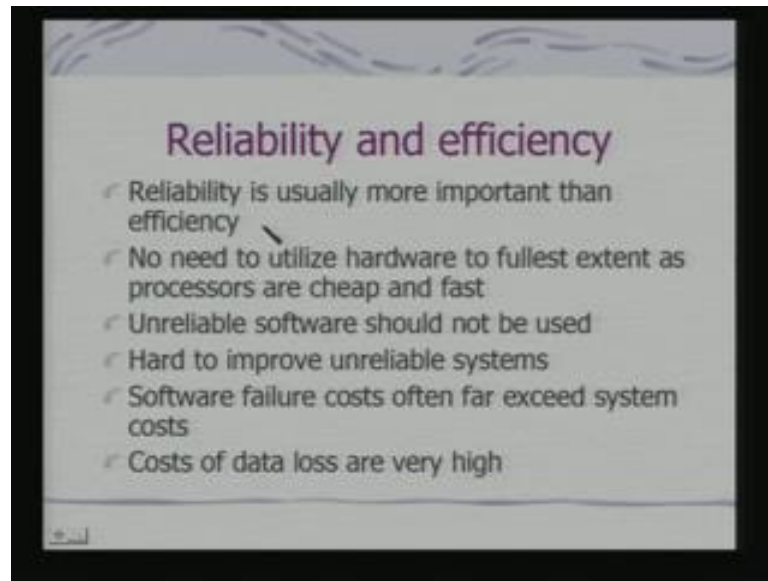
(Refer Slide Time: 13:41)



As reliability increases, system efficiency tends to decrease, because to make system more reliable redundant code must be included to carry out say run time checks this is an example. So, I may like to do a run time memory check o make sure that some bit in the memory has not got arbitrarily flit and producing an error. So, if I can find that there is an error, I can fact that condition earlier. So, these kinds of check codes can tend to slow down the system.

In fact, this would lead to more memory requirement as well ok. So, reliability and efficiently may be at cross purposes. So, that is why the issue comes in if I really want reliable system then to take care of the efficiency, I might need to use additional processing capability. In fact, the basic philosophy wise we say reliability is usually more important than efficiency provided, I meet the constraints.

(Refer Slide Time: 14:42)

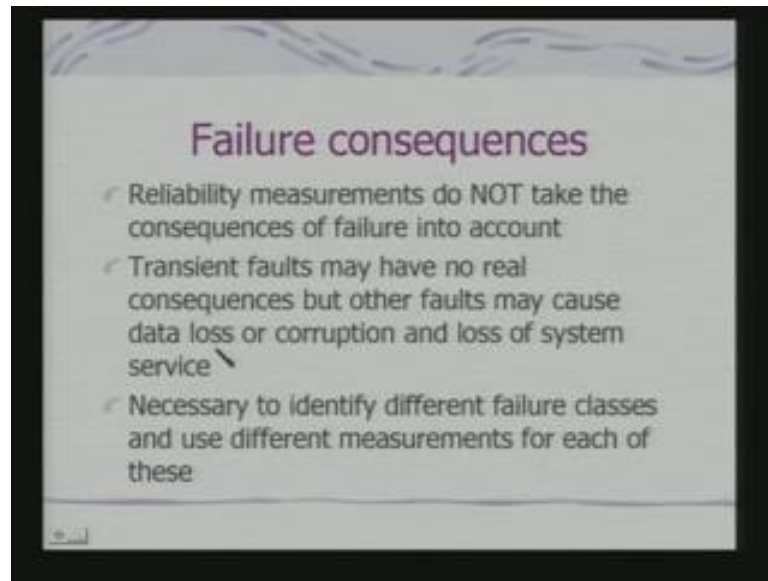


When you are looking at the design, we start with functional specification as well as non functional specifications non functional specification is specify the constraints. So, I might like to have the design, which satisfy those constraints and at the same time reliable may be sacrificing efficiency to a certain extent. So, we say that: no need to utilize hardware to fullest extent as processors are cheap and fast, unreliable software should not be used. And it is difficult to improve unreliable systems.

And software failure costs often far exceed system costs. In fact, and cost of data loss are very high. In fact, if you see the factor which in many cases lead to failures are actually software failures. Hardware failures are there, but, software failures are more critical why because, in many cases you are using a reusing a hardware components so; that means, hardware component has gone through a various process of testing and validation and you are developing a software, which is your application or appliance specific.

So, that software although you might be using some of the components designed earlier, but, that software as a whole might not have been tested. And the second point which comes in its that you are actually, running the hardware under that software control the hardware itself may be reliable, but, under the software control it may lead to some kind of failures ok. So, software failure becomes a very critical component. And we would not like to; obviously, have a data loss because cost of data loss is high.

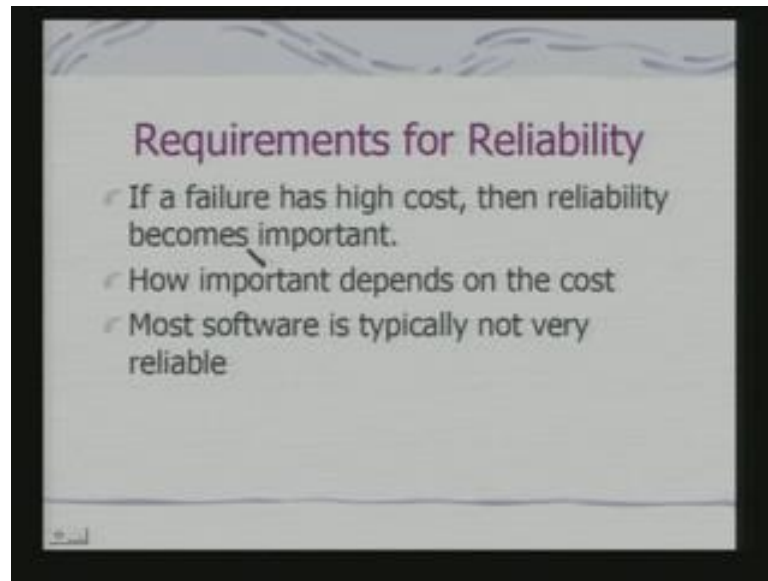
(Refer Slide Time: 16:44)



What are the failures consequences reliability measurements do not take the consequences of failure into account ok. That is we take care of the facts, there would be failure, but, what would be the consequence of the failure that is not taken into account; that means, if there is the failure of on fly by wire system in air graph. The cost of the failure is loss of human lives, but, a reliability measure quickly does not take care of that cost or consequences of failure.

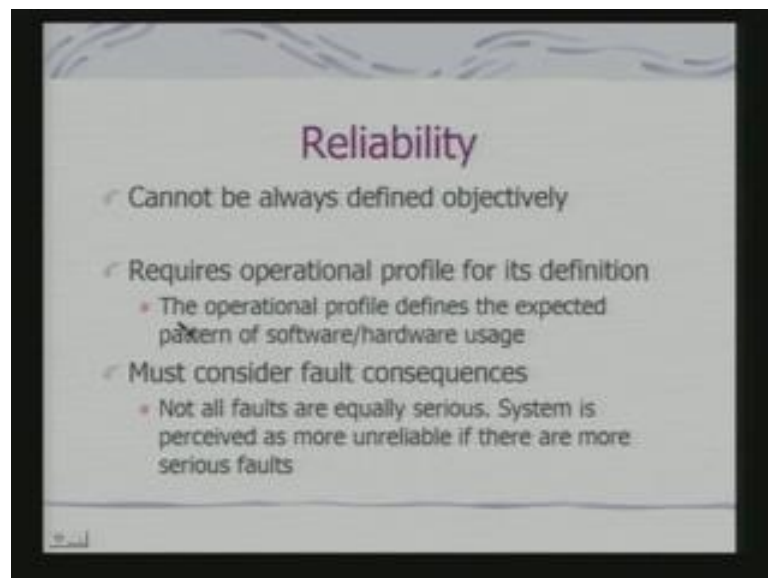
In fact, transient faults may have no real consequences, but, other faults may cause data loss or corruption and loss of system services. So, what is the cost of this lost is not really part of the reliability. So, therefore in order to take care of the cost you can understand, we have to talk about a different measure. And it is necessary to identify different failure classes and use different measurements for each of these if we are talking about the reliability measures.

(Refer Slide Time: 17:52)



So, what are the requirements for reliability if a failure has high cost then reliability becomes important how important depends on the cost and most software is typically not very reliable. Because is the software a basically developed in an application specific fashion and they are not really reliable. And so, the point is if I am really designing for reliability; if I know what the cost of the failure is, I would like to eliminate or minimize those failures for which the cost would be really high.

(Refer Slide Time: 18:27)

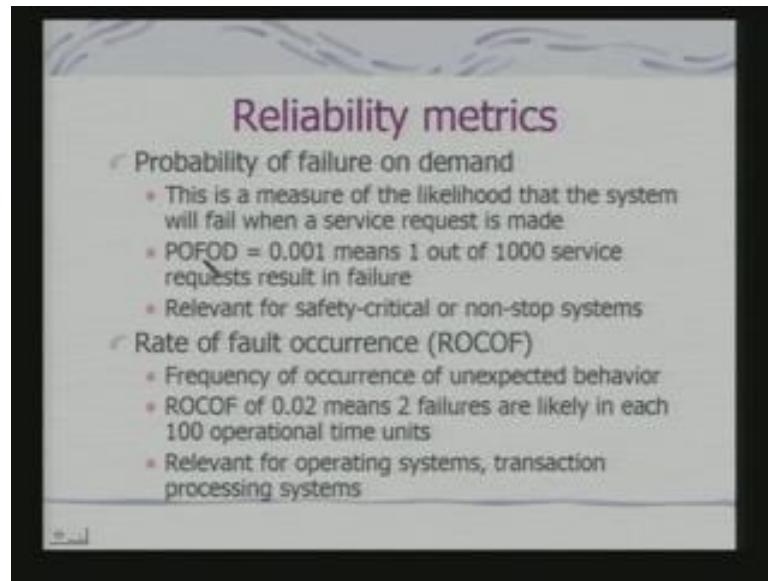


So, in fact the reliability in a way cannot be always defined objectively because, it varies from perception of perception because, perception can change the cost of the failure. And requires operational profile for its definition why the operational profile defines: the expected pattern of software or hardware usage. And from that expected pattern of hardware and software use you can actually figure out whether, these faults should occur what is the probability of those failures to occur and what would be the cost of those failures.

Because, that also depends on the usage patterns say for example, and taking a software example: I might have written a word processor which is got thesaurus service and there is a bug in a thesaurus service and you say that the thesaurus is being really used. If thesaurus is being really used how do I know that only from the operational profile of the software ok. So, thesaurus bug would have less probability of occurrence and if, I want to have release of software with the deadline i might like to have go for the first release with that bug not being removed.

The system is faulty through, but, you can have reasonable reliability of the system because thesaurus would be really used. And we would also consider fault consequences if thesaurus suggests a wrong word at least, the user may use his knowledge and correct it. So, the consequences may not be critical. So, not all faults are equally serious system is perceived as more unreliable, if there are more serious faults. Now, you can understand this all entire discussion is based on the fact and the assumptions, that there would be faults and there would be failures the question is what based we can do after accepting this reality.

(Refer Slide Time: 20:33)



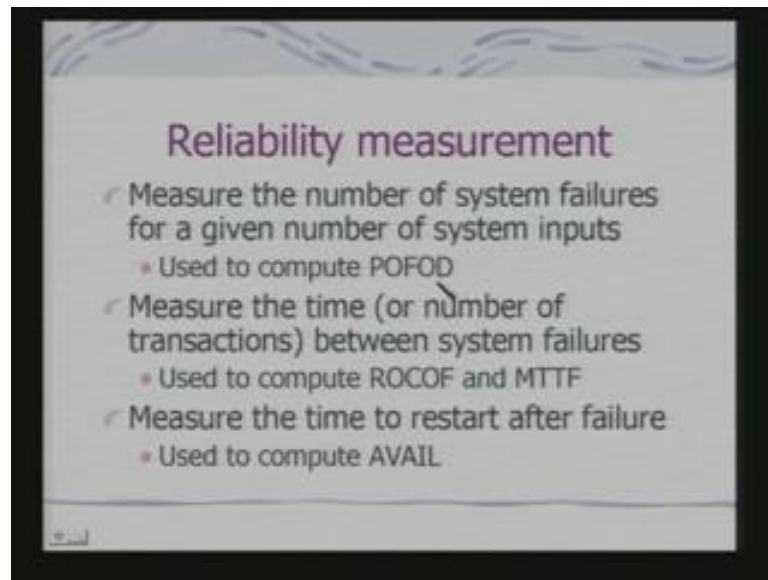
So, what are the reliability metrics first is probability of failure on demand. This is the measure of the likelihood; the system will fail when the service request is made ok. So, we indicate this as POFOD probability of failure on demand see if, I say that this has got 0.001 means; that 1 out of 1000 service requests result in failure ok. It is relevant for safety critical or nonstop systems that; system which are always working and there and it is processing a service request continuously then this becomes important because you are requesting for the service.

If there is a failure; that means, the service is being denied to you. Next thing is rate of fault occurrence this is the frequency of occurrence of unexpected behavior, what is the difference between the 2. In this case it is the failure when you are demanding a service say for example: I am asking for a cash to be delivered through my ATM machine ok. And the system face to deliver the cash fine that is the failure on demand and there would be a measure indicating value. And what is the rate of fault occurrence.

That is the frequency of occurrence of unexpected behavior you go to an ATM machine and you find that it is not working for some reason it is not that you have demanded a service of cash that is cash being deliver to you, but, it is not working it is a random fault which has taken place unexpected behavior. So, it means; that in this case 0.02 means; the two failures are likely in each 100 operational time units. So, this measure is coming over the period of time; for which the system is in operations.

Here, it is measured in terms of number of times that the service is being requested. So, rate of fault occurrence would be relevant for say operating systems transaction processing systems and various other things as well ok. So, these are two basic reliability metrics.

(Refer Slide Time: 22:51)



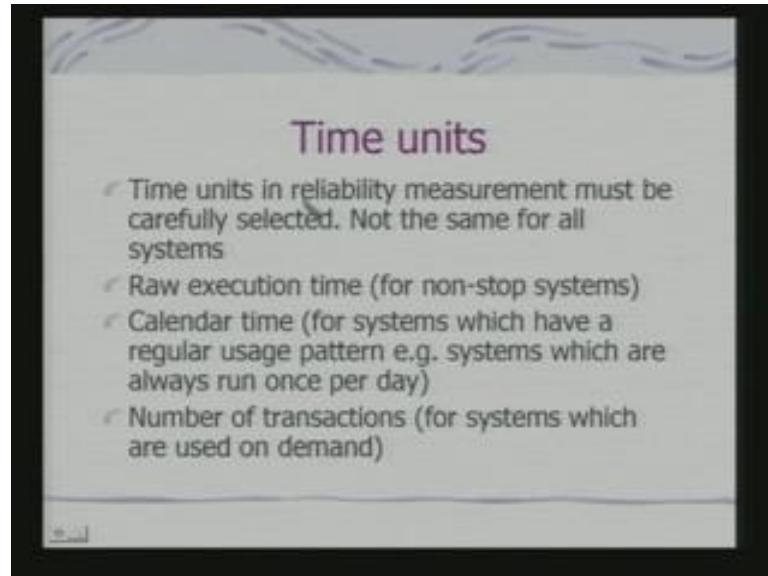
Then how do you measure this; measure the number of system failures for a given number of system inputs ok. So, this is used to compute this failure on demand measure the time or the number of transactions, between system failures and this is used to compute the rate of fault occurrence as well as mean time to failure ok. And measure the time to restart after the failure. In fact, this will lead to another measure, which is called availability and which is link another aspect of your dependable design.

So, what is the basic difference in this case? So, if we give a number of system inputs that means, when we are asking for service we are giving system inputs. Given the system inputs whether, the system is able to provide the service that exactly used to measure this fine. And here you measure the time between the system failures. System has failed today and I have observed the next failure after 3 days then what is the time between the system failures ok. That is used for measuring these parameters.

Now; obviously, this measurements it had to be obtained this has to be obtained at a validation or a testing phase as well as this information's are provided from field data. That is once you have deployed the system from the field data you get this information

once you get this information then, you may think of improving the design at a later version. So, that you get a better reliability.

(Refer Slide Time: 24:45)



There can be various time units, which are used in reliability measurements. So, and we say the time units have to be carefully selected. There can be raw executions time for non-stop systems. It can be calendar time for systems which have a regular usage pattern examples system, which are always run once per day; the number of transactions for systems which are used on demand. So, all this could be used as measure of time because, time becomes a critical parameters to used for computing reliability of a system.

(Refer Slide Time: 25:20)

Reliability

$Rel(t) =$ Probability that the system will operate correctly in a specified operating environment up until time t

Mean Time To Failure
 $MTTF = \text{Expected Value}[Rel(t)]$

- Note that t is important
- If a system only needs to operate for ten hours at a time, then that is the reliability target

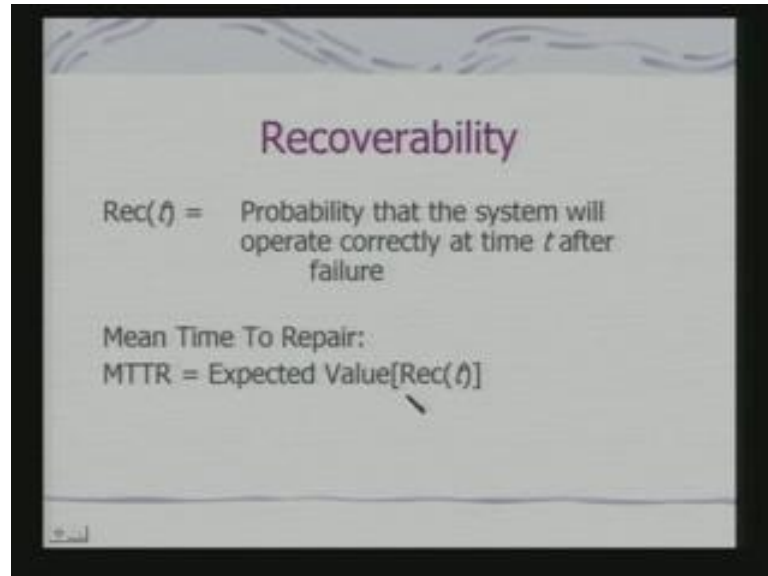
So, formally how do you define reliability? Probability that the, system will operate correctly, in a specified operating environment; up until time t . So, this is what your reliability t is. So, what it says probability that the system will operate correctly in a specified operating environment up until time t ok. So, system is up can working properly until time t what is the probability of time? So, if I have measurements available form different installations.

If I am using the field data using that installation data I can make as assessment of reliability, if I have put to validation testing a number of this systems that I have designed from they are also I can compute this probability and this probability, would be more in terms of relative frequency interpretation of probability I am not really subjective interpretation of probability. And what is the mean time to failure it will be the expected value of reliability measure that we have already defined.

So, it is an expected value of this probability ok. In fact, it is important to note t . So, what we say that if a system only needs to operate for ten hours at a time then, that is the reliability target. We would like to see what is the probability? The system will not failed, if we keep the system running at a stretch for ten hours ok. So, the t now, becomes 10 because it is no point trying to work it out for three 65 days because after running for 10 hours system will be put to off. And again put it up and expected to run continuously for 10 hours.

Related to this is the concept of what is called recoverability probability that the system, will operate correctly at time t after failure ok.

(Refer Slide Time: 27:19)



The slide is titled "Recoverability" in a purple font. Below the title, it defines $Rec(t)$ as the probability that the system will operate correctly at time t after failure. It then defines Mean Time To Repair (MTR) as the expected value of $Rec(t)$. A small black arrow points to the term "Expected Value" in the MTR definition. The slide has a light blue background with a decorative wavy pattern at the top and bottom.

Illustrated distinguish, between the reliability t and recoverability t recoverability is probability the system will operate correctly at time t after a failure. And these can be used to compute what is called mean time to repair. And the mean time to repair is expected value of this recoverability; that means, there is the failure and after the failure you can repair the system and it will operate correctly ok. So; obviously, if a system is not maintainable then you cannot have good recoverability measure.

(Refer Slide Time: 28:13)

Availability

$A(t) =$ Probability that the system will be operational at time t

$E[A(t)] = \text{MTTF} / (\text{MTTF} + \text{MTTR})$

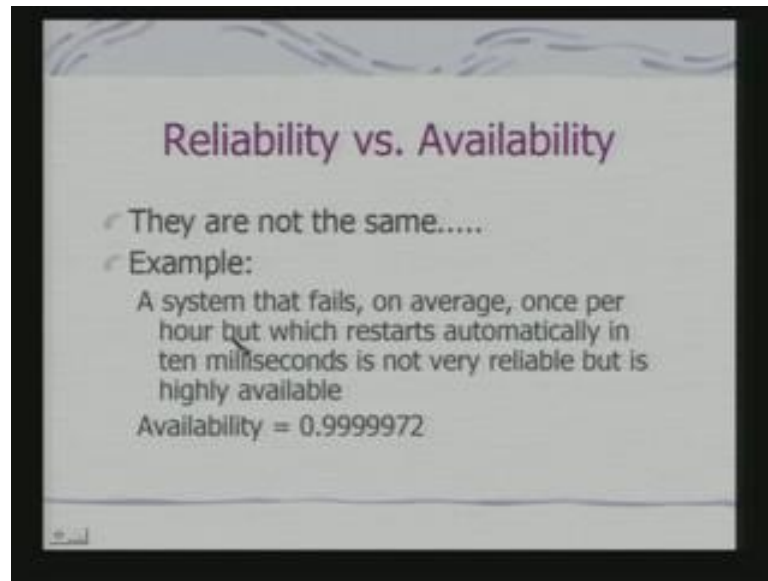
- Literally, readiness for service
 - Only applies when you ask for a service
- Admits the possibility of brief outages
- Fundamentally different concept than Reliability

Obviously since, we are talking about recoverability we come to what is called availability. Availability is the probability the system will be operational at time t ; please try to distinguish between recoverability and availability. In recoverability we are telling that probability that the system will work correctly after a failure at time t . And availability is probability that the system will be operational at time t and. In fact, you can contract with the concept of reliability t also reliability is reliability t is what the probability that the system works correctly for time t .

This is what we are telling probability that the system will be operational at time t respective of what has happen in the past this something very important the distinction between reliability recoverability and availability. So, the expected value of availability is given by this mean time to failure divided by mean time to failure plus the mean time to repair. Literally therefore, availability indicates the readiness for service ok.

That is only applies when you ask for a service and it admits the possibility of brief outages; obviously, because you are accommodating for mean time to repair and you can understand this is fundamentally different concept than that of reliability what reliability is telling reliability tells you that whether the system will work in a proper fashion for time t without a failure here, what we looking at whether the system is available that the system is operational time t whatever has happen in the past ok.

(Refer Slide Time: 30:11)



And that is how these parameters are defined. So, let us take an example a system that fails on average once per hour, but, which restarts automatically in ten milliseconds is not very reliable, but, highly available ok. It can restart if there are restart able system. And for restart it takes just ten milliseconds then if I randomly try to find out whether the system is available or not, I shall figure out that system is really available, but, why the reliability is low because, it is failing once per hour.

So, availability and reliability are not same and depending on application you would like to have may be availability or reliability or may be both.

(Refer Slide Time: 31:05)

Design Tradeoffs

$$\text{Availability} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}}$$

How to make availability approach 100%?

MTTF → infinity (high reliability)
MTTR → zero (fast recovery)

So, the design trade off is we look at availability. So, MTTF by MTTF plus MTTR and the question is how to make availability approach hundred percent. So, there are two options see if, I have mean time to failure approach infinity my availability will be high ok. And if we have mean time to repair going down to zero then also the availability will be high. So, in this case when it reduces we accept the fact that there will be failures or outages perform outages system will recover very fast and that is why the availability will be high. On the other hand we are designing a system to be in such a way that the failure probabilities at very low ok.

(Refer Slide Time: 31:57)

Maintainability

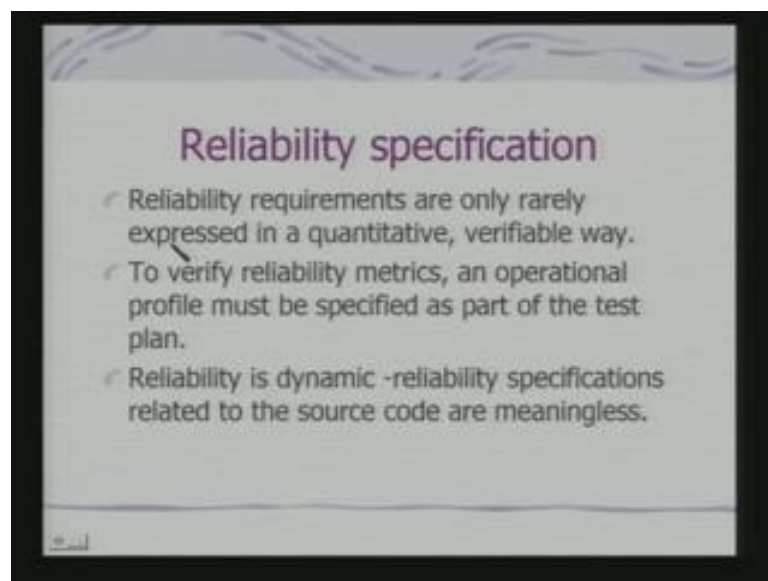
Ability to undergo repairs and modifications

- Maintenance
- Evolution
- Composition
- Manageability

Obviously related to this is the concept of maintainability the ability of the system to undergo repairs and modifications. So, if it is a maintainable then you can realize that availability will also increase because; that means, after a failure the system can be repaired very fast. So, we have got the concept of maintenance evolution composition and manageability. Evolution in the sense that if there are upgrades and modifications.

So, I can use that to evolve or make the system better composition is whether, you can add on a feature add on a hardware component or a software component. And whether you can managed the overall system see for example, if I have battery power device and I really, can have the battery state measured and communicate over a network to a service station then what I am having. I am having a better manageability of the system because, I can replace the battery before the total outage takes place next question is that for a given system how do you specify the reliability.

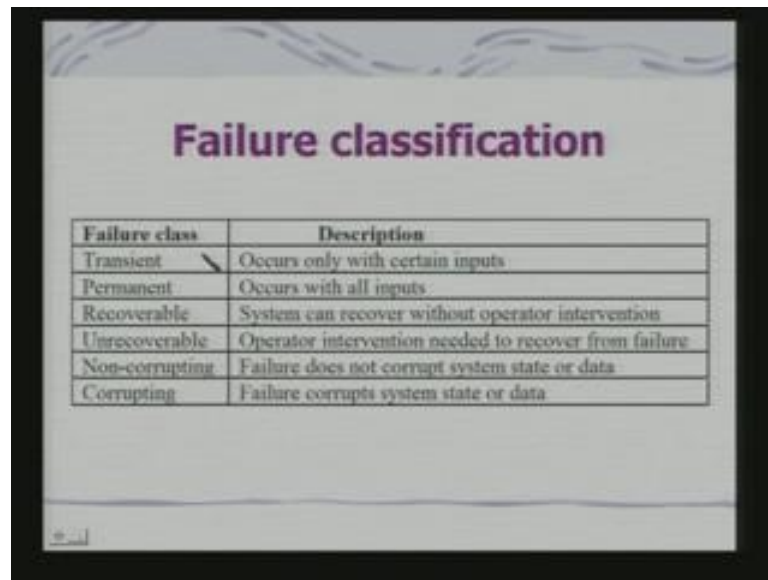
(Refer Slide Time: 33:05)



So, we have understood the basic facets how this specification can be done. Reliability requirements are only rarely expressed in a quantitative verifiable way, but, we need to specify them, to verify reliability metrics an operational profile must be specified as part of the test plan. And reliability is dynamic reliability specifications related to the source code are meaningless. Because this is why it is meaningless because, you have to look at the reliability of the code in the context of the hardware on which it is running.

It cannot be specify independent of the complete hardware because you are talking about on the reliability of the system as a whole in a dynamic work environment. And these reliability specifications can come from or it specifies with respect to different failure classes, what are the different failure classes.

(Refer Slide Time: 34:05)

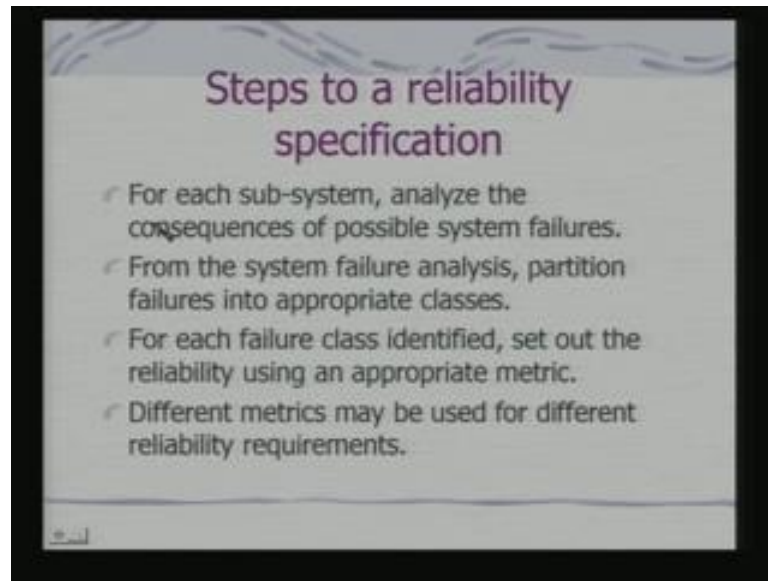


Failure class	Description
Transient	Occurs only with certain inputs
Permanent	Occurs with all inputs
Recoverable	System can recover without operator intervention
Unrecoverable	Operator intervention needed to recover from failure
Non-corrupting	Failure does not corrupt system state or data
Corrupting	Failure corrupts system state or data

Transient occurs only with certain inputs and you can change permanent occurs with all inputs. Then Recoverable system can recover without operator intervention. Unrecoverable operator intervention needed to recover from failure. Then there can be non corrupting failure does not corrupt system state or data. And corrupting failure corrupt system state or data ok.

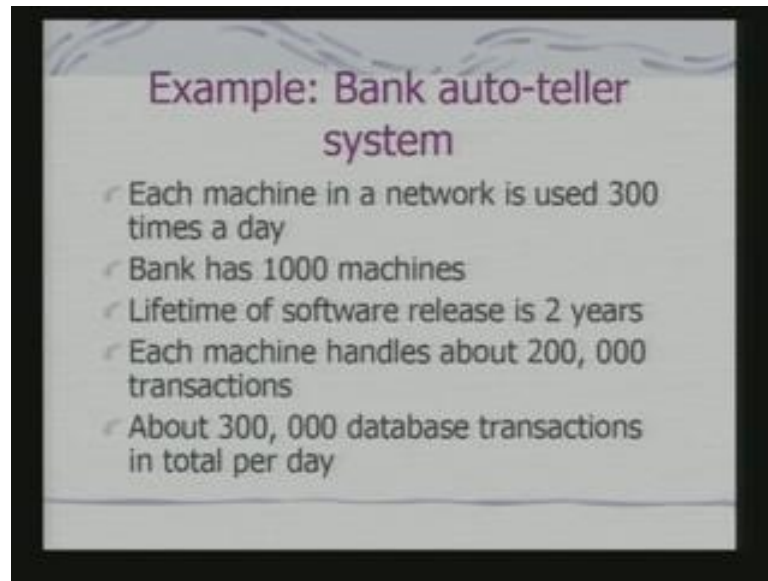
Now, these are the different failure classes and if you are talking about reliability specification you might like to specify reliability specification, with respect to failures belonging to these different classes, depending on the appliance that you are considering.

(Refer Slide Time: 34:57)



So, for each sub system analyze consequences of possible system failures from the system failure analysis partition failures into appropriate classes. For each failure class identified set out the reliability metric using an appropriate metric, different metrics may be used for different reliability requirements. So, this is the whole process of specifying reliability with, respect to a design we have looked at what we have looked at so, for functional requirements non functional requirements. Now, we are looking at reliability specification for the purpose of designing a dependable system or reliable system. Take an example how do you specify that? so, bank auto dealer system. So, each machine is in a network is used say three hundred times a day.

(Refer Slide Time: 35:48)



Bank has 1000 machines lifetime of software releases 2 years that is every 2 years; software gets upgraded, each machine handles about 2000 transactions. And about three hundred thousand database transaction in total per day. Now, we are considering what the overall system, I have got ATM machines ATM machines connected via network ok. Now; obviously, I cannot talk about reliability of a single ATM machine without considering the network, with which it is connected.

Because of failure is not only, because of the failure of the system, but, failure of the server as well as network communication. So, that is why when we are talking about reliability specification of bank auto teller system. We have to take the whole system perspective in mind and specify.

(Refer Slide Time: 36:38)

Failure class	Example	Reliability metric
Permanent, non-corrupting	The system fails to operate with any card which is input. Software must be restarted to correct failure.	ROCOF 1 occurrence/1000 days
Transient, non-corrupting	The magnetic stripe data cannot be read on an undamaged card which is input.	POFOD 1 in 1000 transactions
Transient, corrupting	A pattern of transactions across the network causes database corruption.	Unquantifiable! Should never happen in the lifetime of the system

So, what are the different failure classes you are likely to encounter? 1 is the permanent non-corrupting here the system fails to operate with, any card which is input software must be restarted to correct failure; that means, it does not work with any input. So, this is the kind it is not transient failure it is a permanent failure and a permanent failure can be non-corrupting ok. That means; the data is not destroyed because, data is stored at may be a server data is not destroyed, but, ATM machine does not work. And what is the reliability metrics.

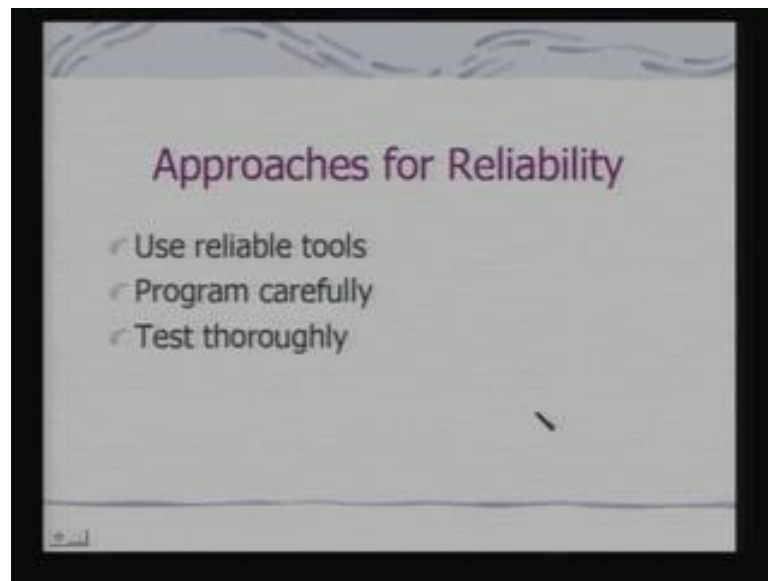
So, you will be specifying this ROCOF, which is related to the failures and not related to transactions. So, 1 occurrence for 1000 days this could be a specifications. Then there could be transient non-corrupting failures. The magnetic stripe data cannot be read on an undamaged card which is input fine. So, this error can come when you are actually putting the card and asking for transactions. So, the major which is being used is POFOD that is on demand service failure.

So, 1 in 1000 transactions and this may be a transient again a non-corrupting fault. And transient corrupting there can be this failures a pattern of transactions across the network causes data base corruption is the dangerous kind of fault ok pattern of transactions which is taking place. So, these error cannot be unquantify. So, I say unquantifiable should never happen in the lifetime of the system because, what does therefore it means. So, once you have the specification it means that your system must be tested on the

software as well as hardware measures should be placed to eliminate this error completely which is not strictly true for other two cases because, the consequence of this failure is not catastrophic ok.

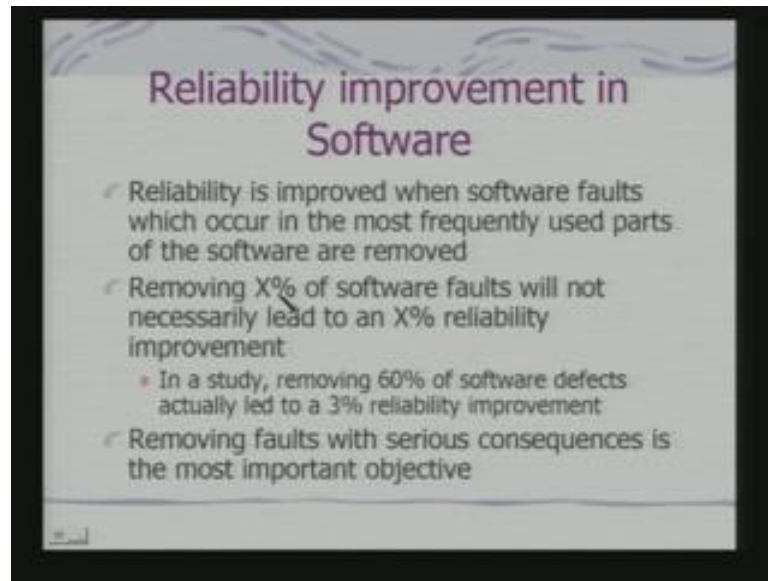
But obviously, you would like on your this failure to occur five times each day ok. So, Then your customers will run away from the bank. So, there that is why I need to have proper reliability metrics specified for this case.

(Refer Slide Time: 39: 04)



So, how do you approach reliability is reliable tools use reliable hardware program carefully and test thoroughly. Now, testing in terms of as a software or hardware would have slightly different implications when you are really looking at reliability, when we are discussed testing, in the last class we just concentrated on functional testing whether it is meeting the functional requirements and also we have talked about some of the performance measures, but, when we are talking about reliability; obviously, that correctness and the functional specification is an issue, but, there are other issues as well.

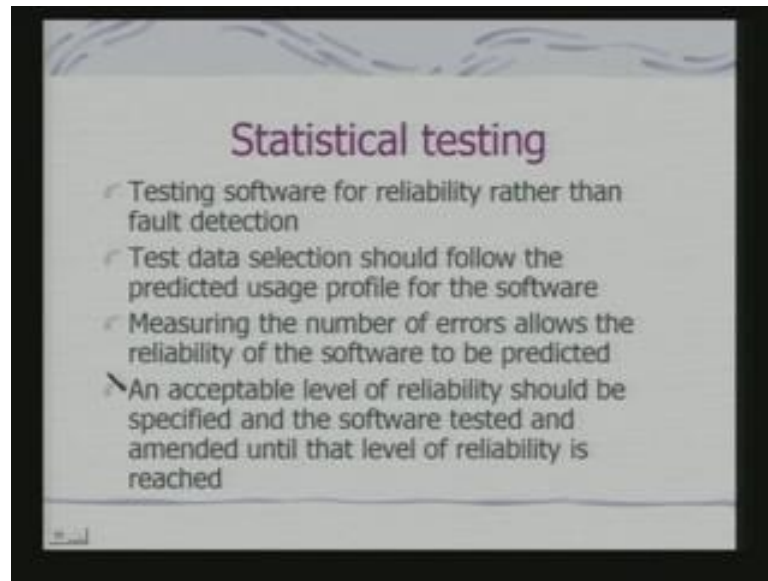
(Refer Slide Time: 39:50)



We say the reliability improvement how it can happen in software because we have found the software is the critical component for a reliability failure in many cases. We see the reliability is improved when software faults which occur in the most frequently used parts of the software are removed. So, removing certain x percent of software fault will not necessarily lead to an x percent reliability improvement why because, it depends on what is the probability with which this part of the software is being used ok

So, it is found there, in a study removing sixty percent of the software defects actually led to only three percent reliability improvement because, there are still box in the software which are part of the commonly used components. And therefore, removing faults with serious consequences is the most important objective for this test purpose.

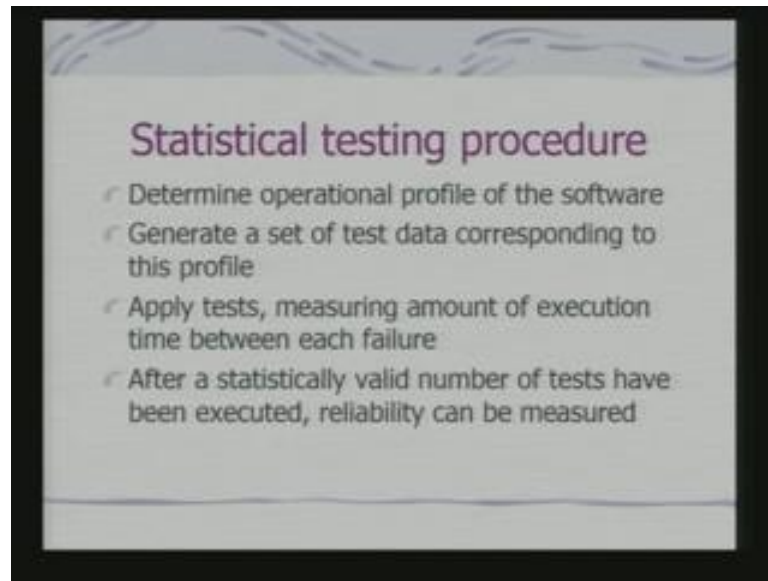
(Refer Slide Time: 40:44)



So, you do what is called statistical testing. The statistical testing for software is primarily objective is for reliability rather than fault detection. So, test data selection should follow the predicted usage profile of the software I give an example of your word processor program. So, there is the certain usage profile which says the thesaurus will be less use. So, you try to have in therefore, more test data corresponding to the more widely used aspects of the software and measuring the number of errors allow the reliability of the software to be predicted.

An acceptable level of reliability should be specified and the software tested and amended until that level of reliability is reached. Because, you cannot remove possibly all box ok. And depending on the user profile you have a tested selection and with respect to the test data try to figure out what is the kind of errors for really encounter whether, that is acceptable or not.

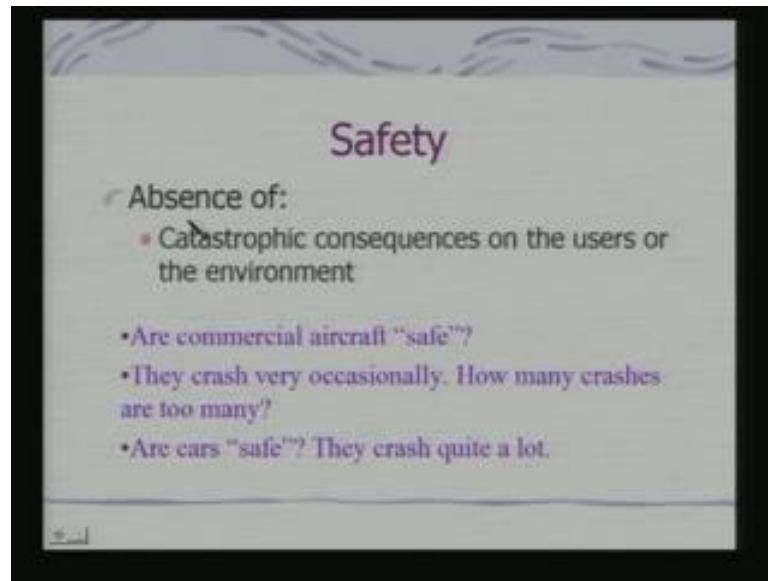
(Refer Slide Time: 41:55)



So, therefore, what is the statistical testing procedure determine the operational profile of the software. In fact, how do you characterize operational profile of the software; if you remember in the last class, we said in a clear box testing there will be various paths in the software which needed to be tested? Now, paths would be associated with different probabilities depending on usage profile of the software. So, depending on the probability I shall select the test data to test out the system.

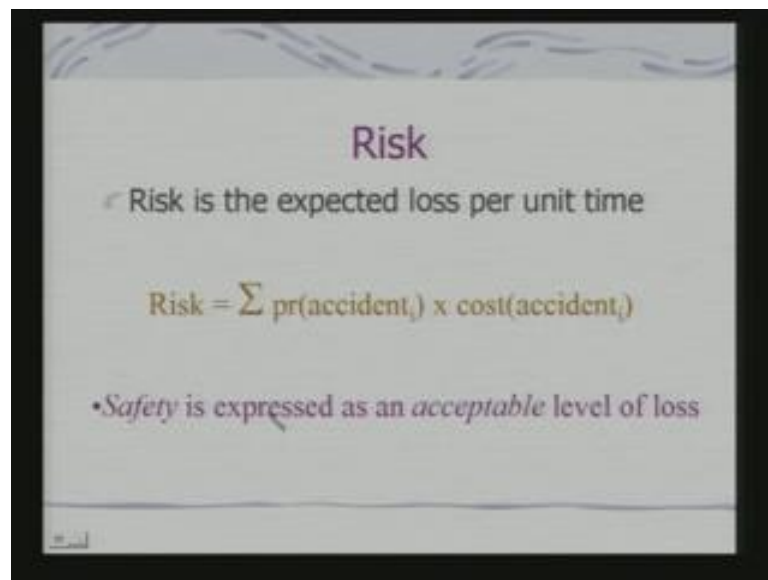
So, apply test measuring amount of execution time between each failure after statically valid number of test have been executed now reliability can be measure. So, I need to have therefore, substantially or sufficiently large number of test inputs to make a kind of a statistical estimate. Next we come to the point of safety.

(Refer Slide Time: 42:46)



Absence of catastrophic consequences on the users or environment is what guarantees safety is. The question is commercial aircraft safe. They crash very occasionally and the question is how many crashes are too many are cars safe they crash quite a lot ok.

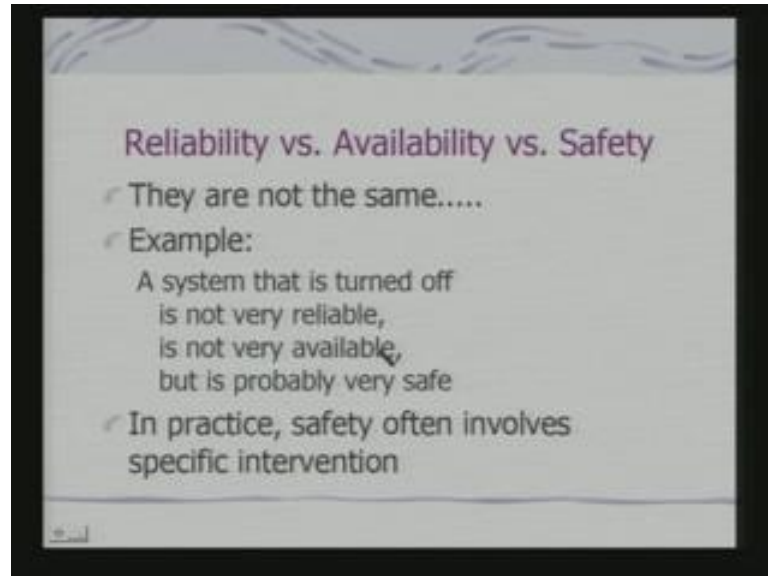
(Refer Slide Time: 43:08)



All these are issues because related to safety. So, what we say risk is the expected loss per unit time. And risk is formally defined as probability of accident into cost of accident. And safety of a system is expressed as an acceptable level of loss; that means,

aircraft will definitely crash we cannot avoid, but, whether it is acceptable or not ok. So, safety is expressed as an acceptable level of loss and risk is measured in this way.

(Refer Slide Time: 43:40)

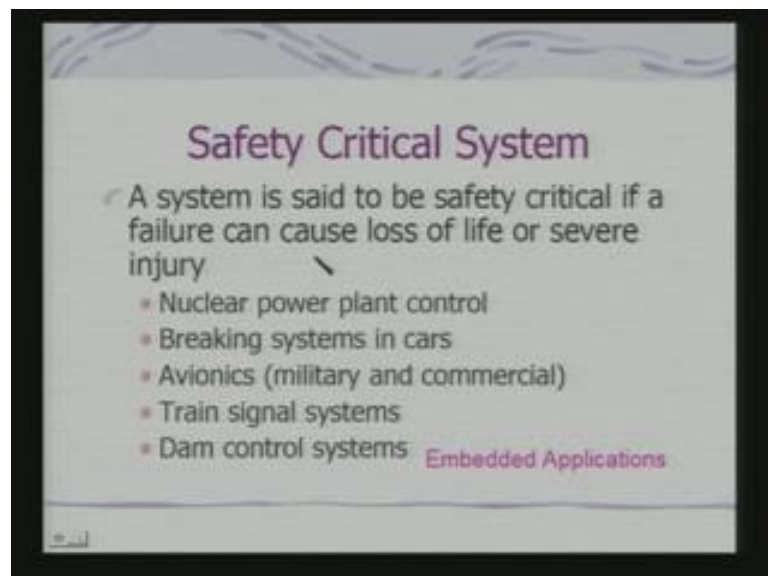


Reliability vs. Availability vs. Safety

- They are not the same.....
- Example:
 - A system that is turned off is not very reliable, is not very available, but is probably very safe
- In practice, safety often involves specific intervention

So, if we now compare reliability versus availability versus safety what we get they are not definitely not same and it is an example. A system that is turned off is not very reliable is not very available, but, it probably very safe. In practice safety often involves specific intervention. That means, that option of the scope has to be built up. And there are number of safety critical systems.

(Refer Slide Time: 44:10)



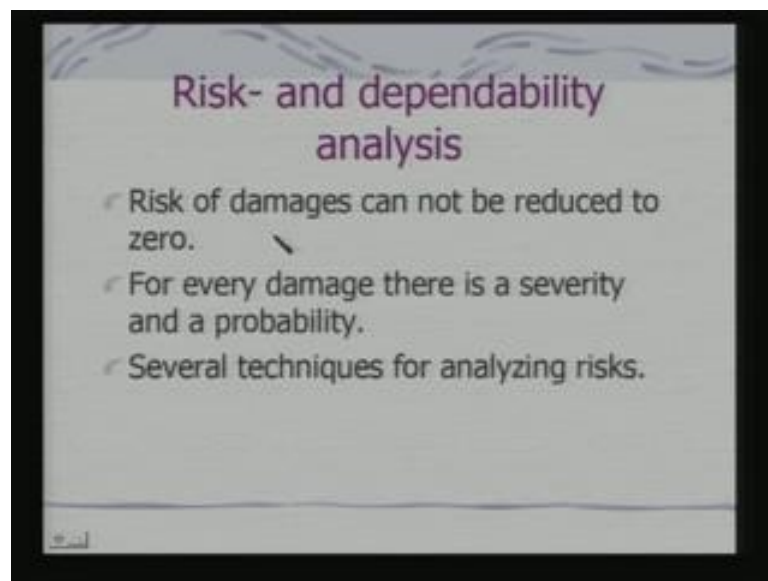
Safety Critical System

- A system is said to be safety critical if a failure can cause loss of life or severe injury
 - Nuclear power plant control
 - Braking systems in cars
 - Avionics (military and commercial)
 - Train signal systems
 - Dam control systems

Embedded Applications

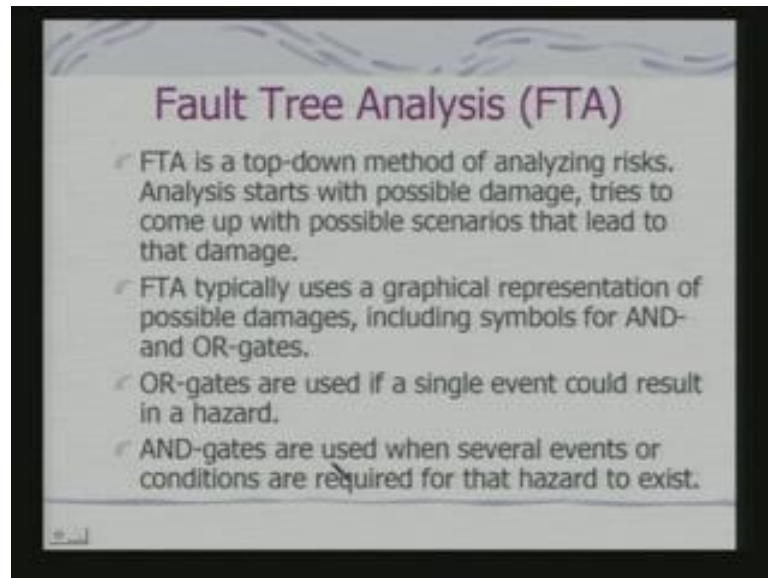
A system is said to be safety critical if a failure can cause loss of life or severe injury. In fact all these are examples of your embedded system hardware and software: Nuclear power plant, braking systems in cars, Avionics, Train signal systems, dam control systems all these are examples of embedded applications. And control related applications, where the safety becomes a key component ok. And the design should take care of the safety as an issue.

(Refer Slide Time: 44:42)



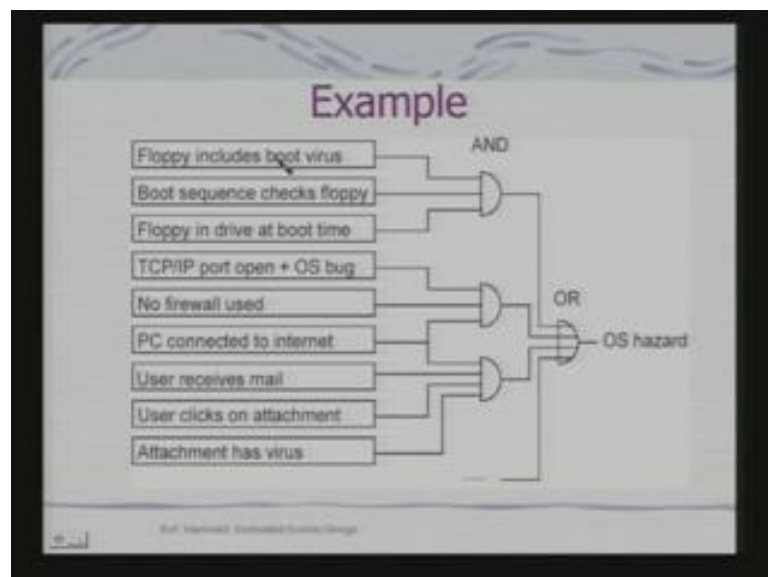
So, in order to do this safety analysis we need to do what is called risk and dependability analysis. Risk of damages cannot be reduced to 0. And for every damage there is a severity and a probability. And we have to use a technique to find out if a fault occurs what would be the final risk that is involved. In depending on that we have to take measures to eliminate those faults. The more common technique is what is called fault tree analysis.

(Refer Slide Time: 45:15)



FTA is a top down method of analyzing risks analysis starts with possible damage tries to come up with possible scenarios that can lead to that damage. And it typically uses a graphical representation of possible damages including basically AND, OR gates. OR gates are used if a single event could result in a hazard. AND gates are used when several events or conditions are required for that hazard to exist. Take an example.

(Refer Slide Time: 45:44)



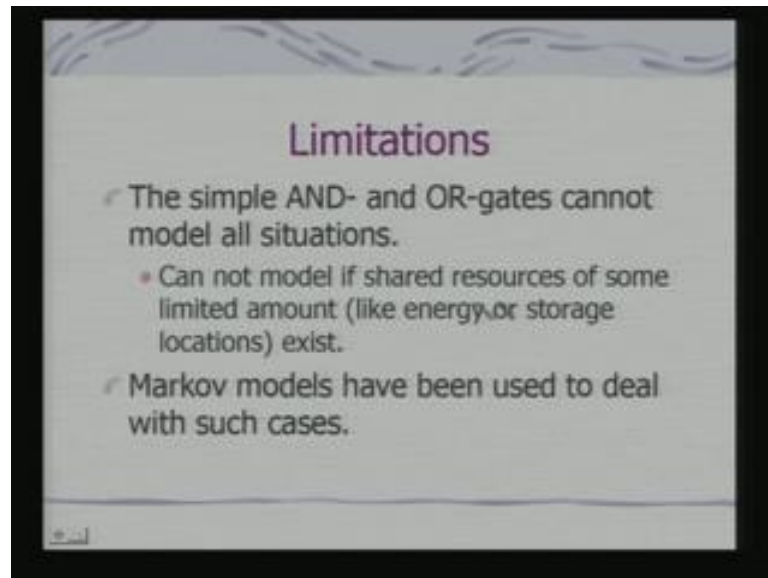
We are talking about OS hazard ok. And what we have provided here the different faults which can actually lead to this OS hazard. And this AND gates means what that floppy

includes boot virus. Boot sequence checks floppy, floppy in drive at boot time. If all these conditions are true then only the virus can cause an OS hazard this is an OR gate which has been represented here can cause an OS hazard. So, if I want to minimize the OS hazard because if OS hazard has got high risk, I would stop or eliminate by the floppy drive from my computer system ok.

So, that is exactly what we say when we say that faulty analysis or a dependability analysis of the system. Because if I want to make system more dependable, I need to do after I have done the design this dependability analysis try to figure out the factors affecting the dependability of the system and minimize the faults related to those factors or eliminate the sources of those faults ok. So, this is an example which is very commonly known to you the faults that you really encounter when you are using PC on a network and how to eliminate the possibilities of these faults and increase the dependability of the system.

Now, the faulty method if you see we are using AND, OR gates and the assumption is there are no dependencies between these 2 faults. If there are dependencies, between these 2 faults or conditions then I cannot really use this AND gates or an OR gate based analysis. This is in a way if you are familiar with the concept of AND, OR graph I am representing the whole system scenario through what is called AND, OR graph. And I am propagating through the AND, OR graph the affect of the particular fault that is from this is the leaf node and from there and trying to find out what is its impact on the overall system. Because OS hazard can get into other gates other nodes leaving to the route node which will be system as a whole

(Refer Slide Time: 47:57)

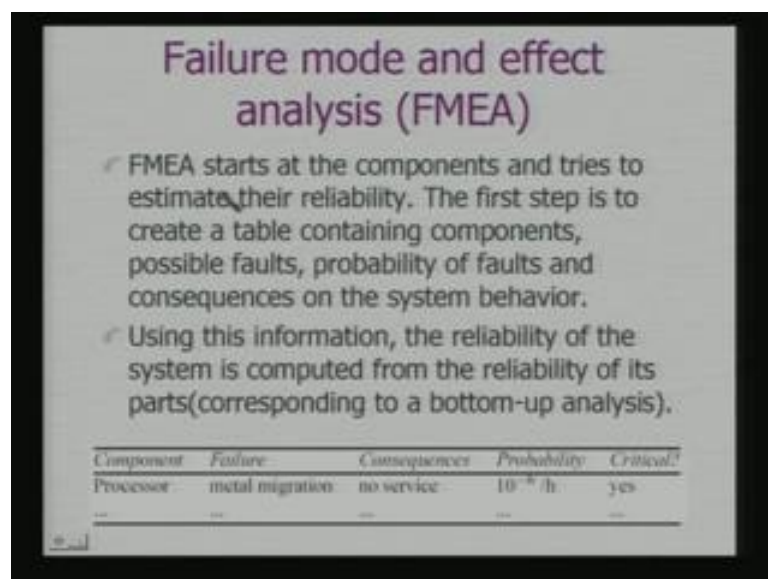


Limitations

- ✓ The simple AND- and OR-gates cannot model all situations.
 - Can not model if shared resources of some limited amount (like energy, or storage locations) exist.
- ✓ Markov models have been used to deal with such cases.

So, what are the limitations the simple AND OR gates cannot model all situations cannot model, if shared resources of some limited amount like energy of storage locations exist linking up to faults or linking up to factors. And there are algorithms making use of Markova models that had used to deal with such cases. We shall not be going into those algorithms it just trying to make you conscious about that these kind of analysis is important for designing dependable systems.

(Refer Slide Time: 48:31)



Failure mode and effect analysis (FMEA)

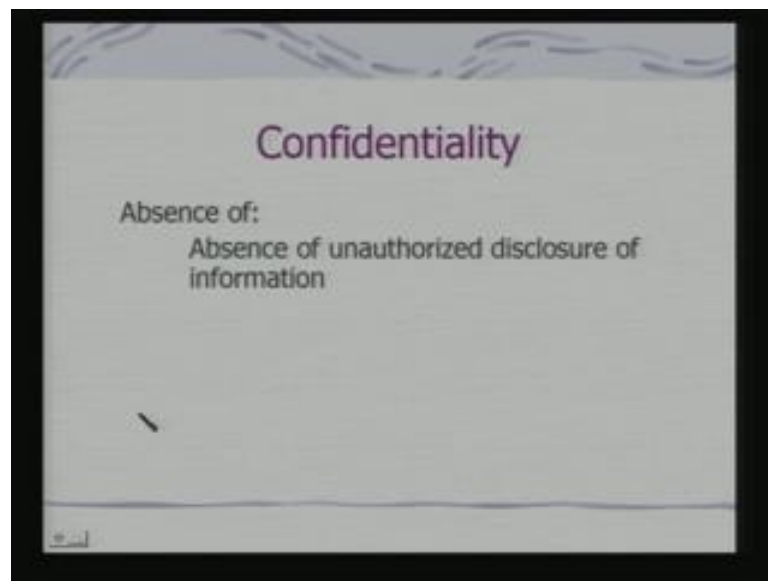
- ✓ FMEA starts at the components and tries to estimate their reliability. The first step is to create a table containing components, possible faults, probability of faults and consequences on the system behavior.
- ✓ Using this information, the reliability of the system is computed from the reliability of its parts (corresponding to a bottom-up analysis).

Component	Failure	Consequences	Probability	Critical?
Processor	metal migration	no service	10^{-6} /h	yes
...

The other thing is failure mode and effect analysis this starts at the components level and builds up the whole system starting from the components. So, start of the components and tries to estimate their reliability the first step is to create a table containing components possible faults probability of faults and consequences on the system behavior ok. Using this information the reliability of the system is computed from the reliability of its parts corresponding to bottom up analysis.

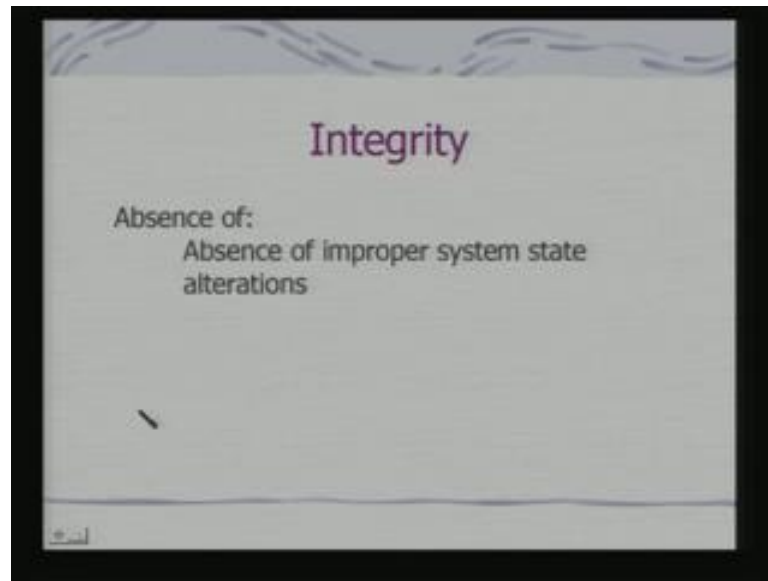
So, takes for example, if this is the processor if there is the failure of the processor consequences no service and the probability is 10^{-6} and it is critical or not it is definitely critical the whole system will fail. So, basically what we are telling is I have got a system, I have got the component with respect to each components; I can have the possible faults of the components probability of the faults that can occur. And what is the consequence; on the system behavior and combining them together, I can compute the reliability of the system ok.

(Refer Slide Time: 49:48)



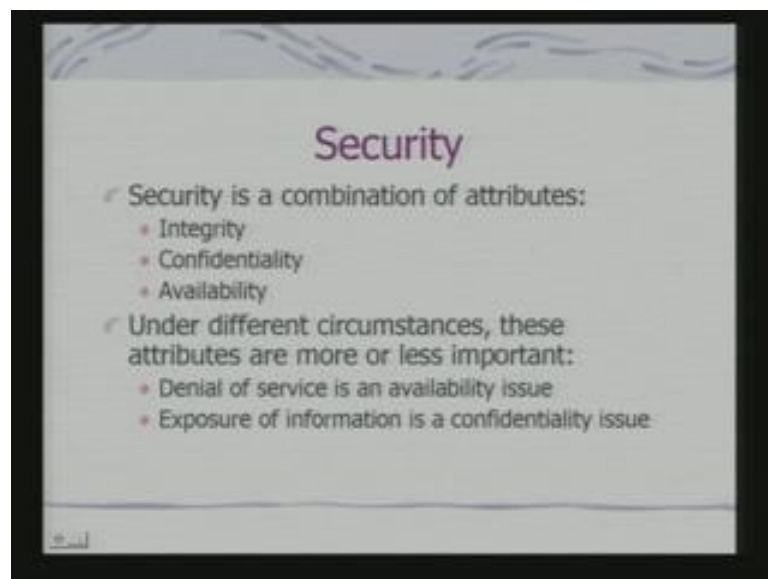
Obviously this can happen once you have done the design then comes the issues of security confidentiality is absence of unauthorized disclosure of information.

(Refer Slide Time: 49:57)



Integrity is absence of improper system state alterations.

(Refer Slide Time: 50:03)



And things put together are what we get security. Combination of integrity, confidentiality and availability is what defines security of a system. Under different circumstances these attributes are more or less important. Denial of service is an availability issue. And denial of service can happen because of in security. And exposure of information is a confidentiality issue. So, what are the security requirements in a today

embedded system which is expected to be connected over the network, one is user identification.

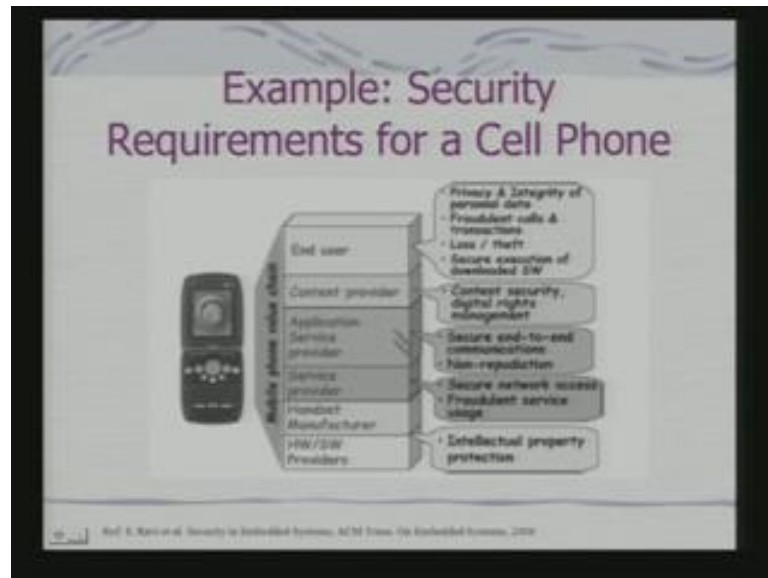
(Refer Slide Time: 50:38)



Now, it is not that all these things would be applicable for everybody, but, these are the general requirements: User Identification, Secure Network Access, Service access if authorized. Secure Communication. Confidentiality and integrity of the communicated data then storage the data store should not be disturbed then, question of content security that is usage restrictions of digital content stored. Question is availability can perform its intended function and service legitimate users at all times.

So, all these are different aspects of security requirements ok. Network access communication storage, content security that is; the data which is currently stored as well as the availability.

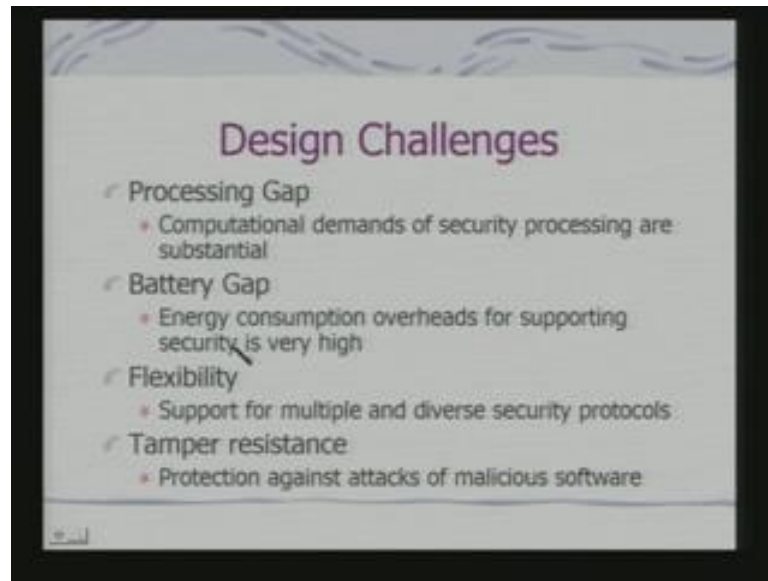
(Refer Slide Time: 51:27)



Let us take an example of a cell phone, which is an embedded system. Now, there is a variety of requirements from the end user's perspective. You would like privacy and integrity of personal data, you would like that no fraudulent calls and transactions are made. And you would like the secure execution of downloaded software. You should not have a cell phone virus disturbing the whole system. Then from a content provider's perspective, content security and digital rights management.

So, it is not that you should download the picture, modify the picture, and start selling that picture as the in a cell phone screen. Then application service provider should provide secure end-to-end communications. Service provider should provide secure network access. Handset manufacturers should have software and form a secure against copying that is intellectual property protection. So, these are security requirements and security requirements coming at multiple levels and that is what to be efficient. Because it is not just with respect to the device alone, it is with respect to the different services that are being used by the device.

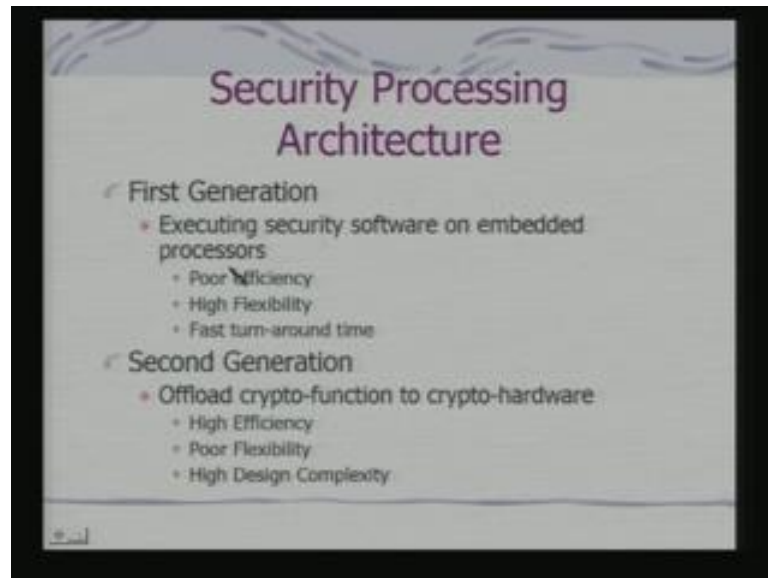
(Refer Slide Time: 52:39)



So, what are the challenges because, computational demands for security processing are substantial, because if you are actually to implement any kind of coding scheme computational requirements will be high your energy consumptions will be high. And there is the need for flexibility, because if you see there are security requirements at multiple levels at a service provider level at manufacturers level and at the level of actually user end user.

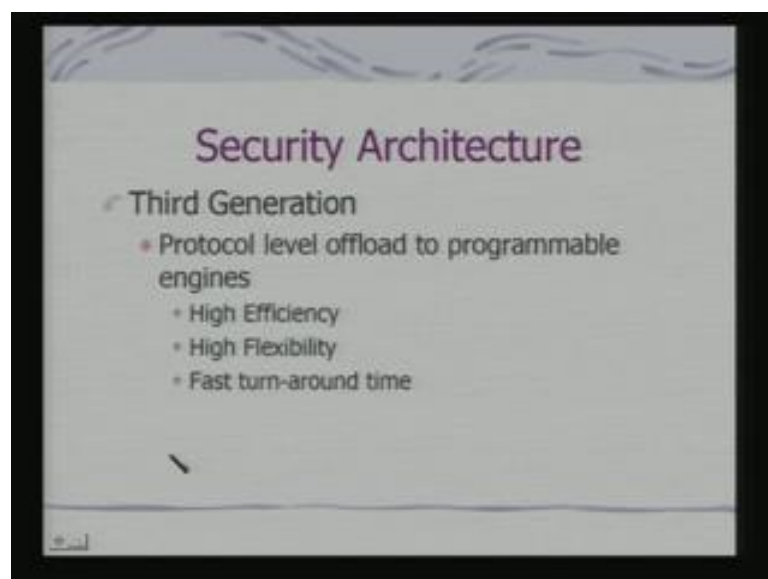
So, at various levels there is a necessity for security and there is a need for therefore, flexibility in the security algorithm being implemented in embedded systems. And it should be also tamper resistance protection against attacks of malicious software. So, what it implies, it implies that since you are using limited processing power and limited battery you need to develop special architectures for this kind of security requirements.

(Refer Slide Time: 53:36)



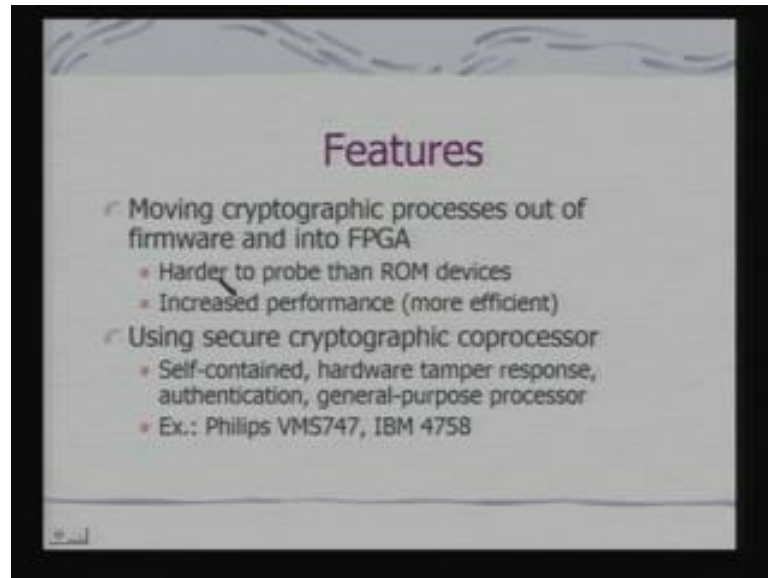
So, there are variety of architecture options first generation architecture is you implement the security algorithm on the basic embedded processor itself. It is flexible, but, less sufficient. Second generation is offload crypto function to crypto hardware crypto function is basically, for encoding the data securing the data. So, here you are start using special hardware it is high efficiency, but, poor flexibility ok. Because you can use hardware with dedicated crypto algorithm.

(Refer Slide Time: 54:05)



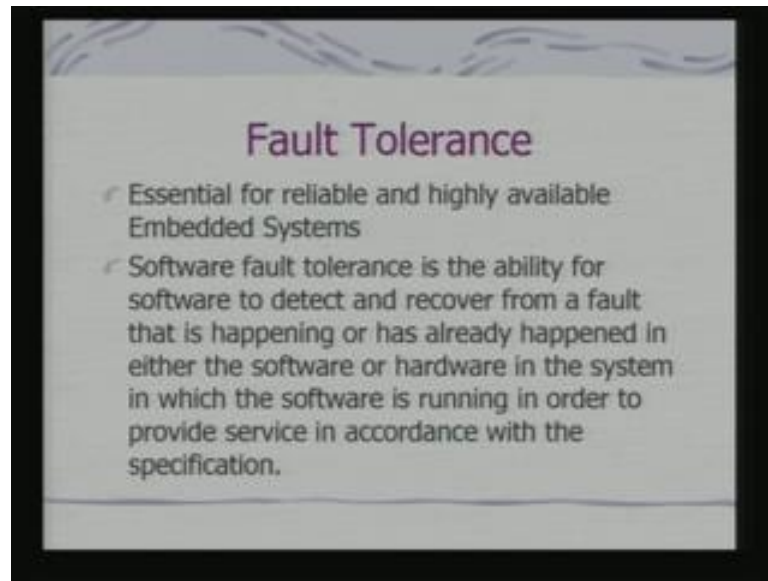
The other option is you offload to programmable engines high efficiency high flexibility and fast turnaround time. That means, you may have an FPGA kind of a thing implementing the crypto algorithm. So, that gives the flexibility.

(Refer Slide Time: 54:24)



That gives you something else as well. It is harder to probe the ROM devices because algorithm gets implemented in the hardware itself. So, by somebody cannot really crack in to your encoding scheme by looking at the code stored in ROM. And you get increase performance. And also there are cryptographic coprocessor which coming from a variety of venders. In fact, o map also has got a version, with cryptographic hardware built into the system to provide this kind of security to a appliances like cell phone.

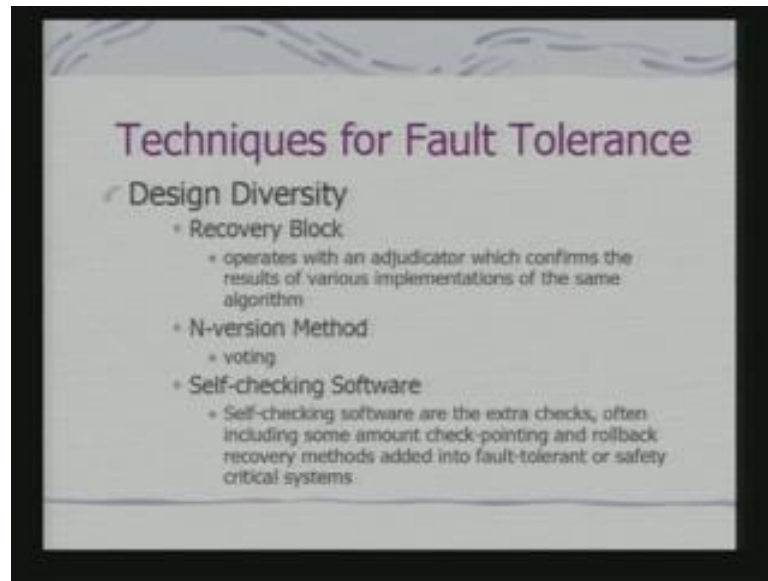
(Refer Slide Time: 54:53)



The last feature, that we shall look at fault tolerance because, fault tolerance also increases reliability by including redundancies. And in fact software fault tolerance is the ability for the software to detect and recover from a fault; that is: happening or has already happened in either the software or hardware in the system. In which the software is running in order to provide service in accordance with the specification.

So, when we are talking about software fault tolerance it is not always redundancies, when we are talking about hardware fault tolerance, in most of the cases you are actually putting in redundancies and this is again important for safety critical systems. So, how you go for fault tolerance very simply; these are the basic issues what we called design diversity ok.

(Refer Slide Time: 55:40)



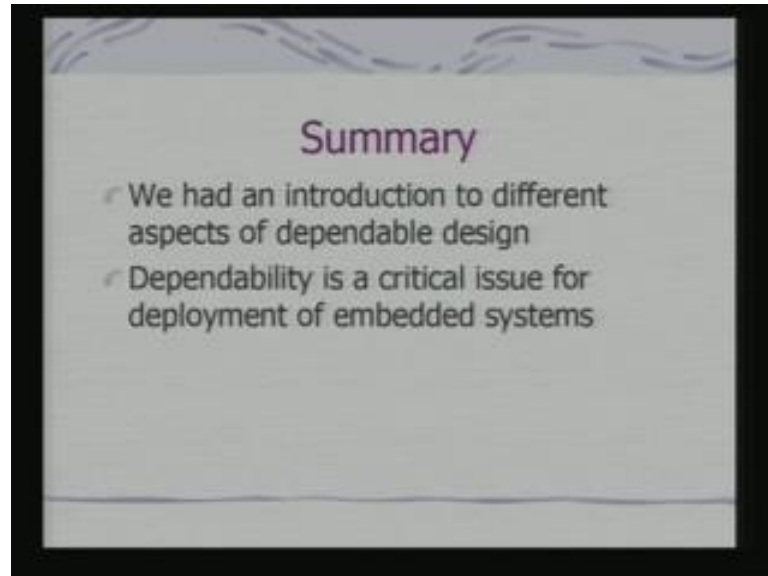
You would design, in such a way that you have a diversity built into the system first is recovery block based approach the system operates, with an adjudicator which confirms the results of various implementations of the same algorithms. There can be multiple implementations of the same algorithm and multiple hardware and an adjudicator basically decides, what is to be done next depending on the output of multiple algorithms related to this is N version method. So, these are all redundancies there are multiple implementations.

In an N version method you really do not have an adjudicator, but, what you have you use voting depending on the maturity operation suggestion you actually tied a step. In fact your all your space shuttles and space operations the ground control of the space shuttles. All of these really used some kind of fault tolerance systems, with N versioning and with multiple systems doing the same job using adjudicator or a voting pattern to make the system completely fail shape.

There is also you can use self checking software; self checking software have extra checks often including some amount of check pointing and rollback recovery methods. If it detects there is the failure, it would roll back certain actions so; that means, it keeps track of what it is doing and that is to make the system fault tolerance on a safety critical systems. So, it would more it will it has the ability to rollback the ability to rollback is

important because, otherwise this action may lead to some permanent failures. So, today therefore, we had introduction to different aspects of dependable design.

(Refer Slide Time: 57:39)



And dependability is the critical issue for deployment of embedded systems. And. In fact, today if you look at some of the challengers of embedded system design dependable design is 1 of the biggest challenge. Today, for designing embedded systems many of the other design issues have reached a certain level maturity dependable design is the one of the issues, which is most challenging in this world of network embedded appliances ok.