

Real-Time Systems
Prof. Dr. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture No. # 02
Real-Time System Characteristics

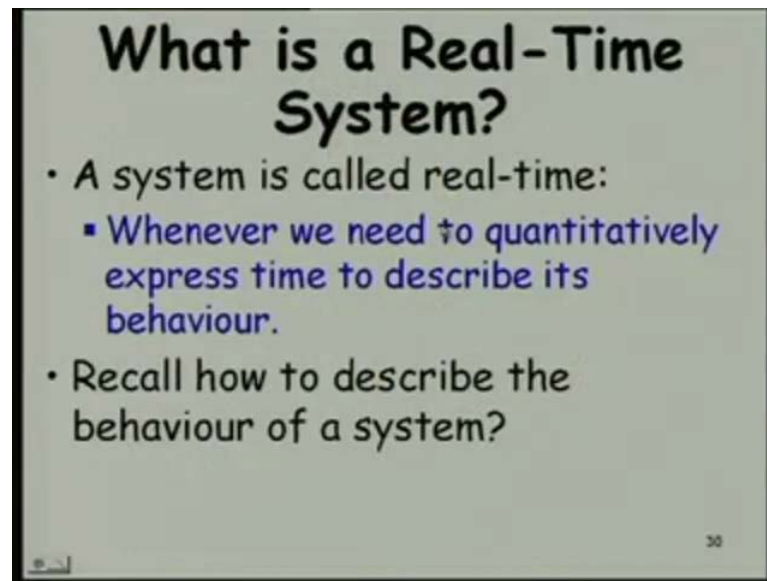
Say good morning, let us get started lecture 2. So, yesterday we had seen some very basic introduction to embedded systems and real time systems, and we had said, that embedded systems constitute majority of the real time applications, but of course, real time systems are not confined to just embedded applications, **application** also elsewhere as we will see later.

Yesterday, we had seen some example applications of real time systems, embedded systems, and we had seen in a basic structure of the embedded systems, and we had said about different types of sensors, actuators, and then this signal need to be conditioned, and then, there is a computer processes, this, human interface to configure and control. So, let me just ask one or two questions before we get started. Just to check that you are following the course. So, can you just name one or two sensors and principle on which these are based? Anybody would like to name just a sensor? **Temperature sensor, which one, temperature sensor, ok. (())**. So, what will be the principle, on which it will be **based**? Thermo couple; thermo couple any other sensor. Light sensor.

Light sensor. So, what will be the under laying principle? **(())** will be converted into that electrical **(())**. So, photovoltaic, **photovoltaic** electricity, so, that is the principle. Pressure sensor, pressure sensor, so, what will be the physical principle behind that? **(())**. How, I mean, you just apply pressure to anything, it would not convert to electricity is not it. So, what is the... **(())**. Piezoelectricity, fine.

So, let us proceed further with our discussion. So, today, we will look at some very basic characteristics of real time systems, so that, when we start discussing about the operating systems, we will keep this in mind.

(Refer Slide Time: 02:36)

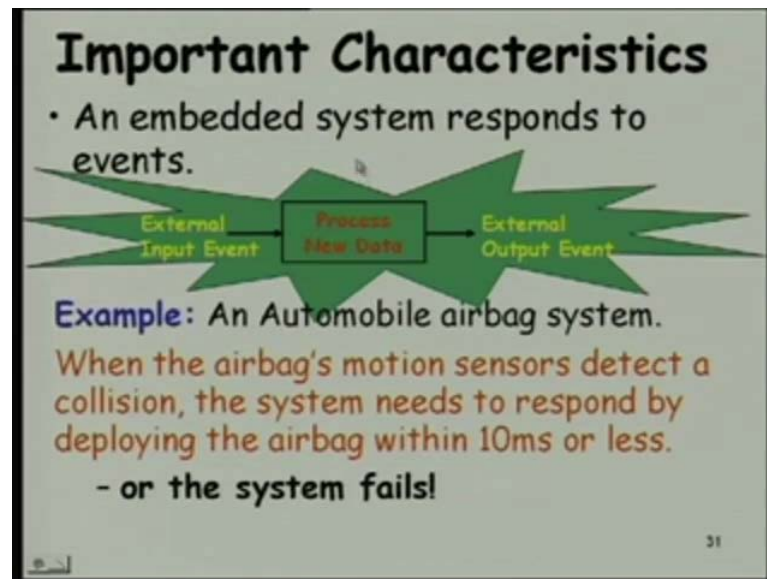


So, I will just have one or two slides which had already seen yesterday, just for continuity. We had said that a system is real time, when we, **when we** describe the behavior of the system, we need to express time quantitatively, but the question is how do we express behavior of any system; suppose, you are ask to describe the behavior of a system, let us say anything may be a mobile phone or something.

So, **you are ask, to describe,** even a system you are ask to describe the behavior. So, what will you do, how will you describe the behavior? **(())**. The behavior is given in terms of the input-output behavior basically. So, what are the functions it supports? The function, each function would basically require some input and then that will produce some output. So, we have to list out all these functions, in the kind of input and output.

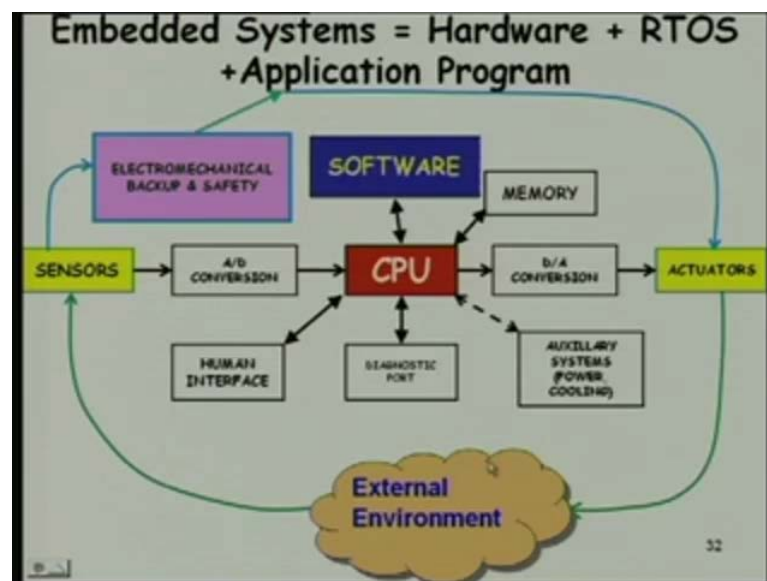
And if such description of the behavior require some quantitative notion of time, we have a real time system, but every function may not be real time, that we have to keep in mind; I thing has we proceed this point will become clear.

(Refer Slide Time: 04:03)



So, this diagram just shows that, **that** there are events that, **that** are recognized by sensors in embedded system and the processor, processes this and produces external output. Just an example is a automobile system, you would see that, whenever the sensor detects, a collision will be processed and what needs to be done, inflating the air bag and rejecting it, will be done in 10 millisecond or less otherwise the system will fail.

(Refer Slide Time: 04:46)



So, yesterday we were just observing this diagram very basic diagram, but we have to keep this in mind as we proceed. So, the kind of systems, that we will be talking of there

will be a CPU, which will be getting data from events from sensors or may be data and converted analog to digital conversion and then read by the CPU. And these are processed by the software and output is produced and the output is converted and drives the actuator, can be a motor or can be a hydraulic pump or whatever and we will have some memory in the system, as I was saying that, in these kind of systems not really have magnetic memory, very rare in embedded system to have magnetic memory.

These are normally semiconductor memory for, but as we know that semiconductor Ram, Ram memory is volatile. So, how do you store permanently data? Rom. Rom is not being used very frequently, now it is getting absolute we have flash memory. (()).

So, flash memory, like your pen drive, you know you have gigabytes of memory available in flash memory. So, what exactly is a flash memory, just to get things, because later we would not see those hardware details, what exactly is a flash memory, for example, your pen drive what is it. (()).

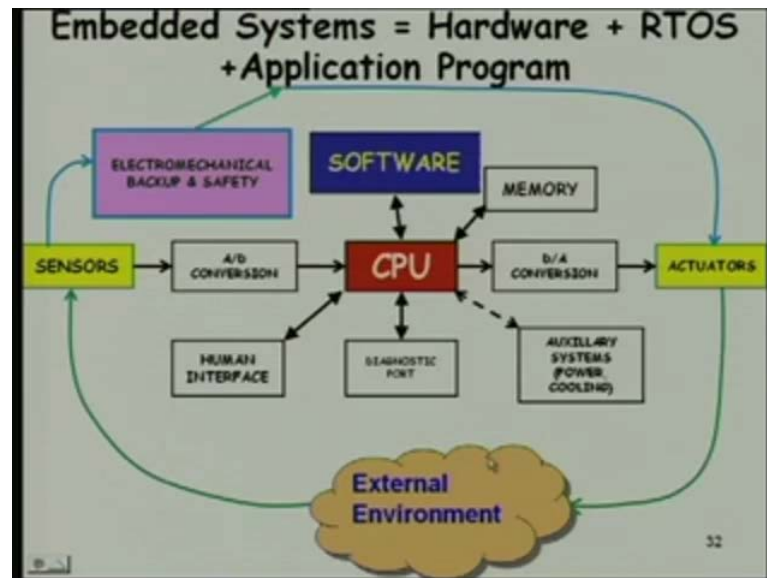
Semiconductor, but what kind of semiconductor, is it a Rom, is it a Ram, is it what, what is it? (())). Sorry, can you please repeat again. Is a permanent memory. Permanent. (()). But how can you get a, permanent memory silicon? (()). It is a type of electrically erasable programmable reader, no worry. So, eeprom, but eeprom existed from long time, we did not have flash memory that time is not it; eeprom is not really flash memory, because these had restrictions under storage, you will write bit wise there and required high voltage.

But in a flash memory, you write in a flash several kilo bytes of data read and written in one operation. So, these have more efficient and the writing occurs at 5 volts earlier in the eeprom, you need a special eeprom programmer which will generate I think 24 volts or something, much more higher voltage was necessary, because tunneling of electrons is required. So, here similar principle, but much more improvised and read and write occurs in large chunks of data several kilo bytes in a flash.

So, this is much more efficient cost effective and of course, lot many developments have occurred, like rather than just storing charge is present or not present the quantum of charge present, that also indicates you know different level. So, one cell can store several bits of data, depending on the levels that are recognized. So, in all these devices, whether

you talk of a camcorder or you talk of a cell phone, flash memory is used as a permanent memory, because these are light weight these are consume less power, less costly. So, these are the reasons why all these devices we look at; whenever we talk of a permanent memory, we will talk of flash memory and of course, it is much faster than hard disk, because these are after all semiconductor memory.

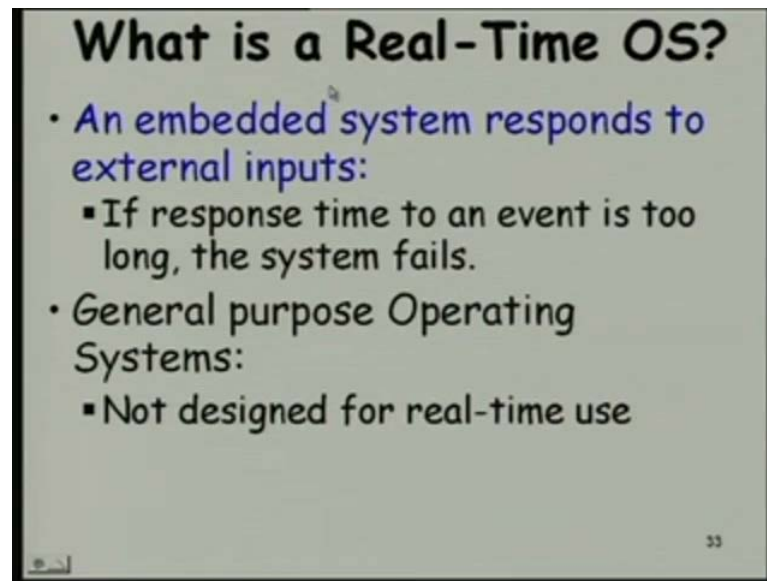
(Refer Slide Time: 09:23)



So, let us pressure do the other blocks in this; so, we have some human interface for configuration, will, will not really bother in this course about this one and then we have auxiliary systems, which are interface to the CPU for controlling power cooling all those things, reporting on a malfunction all those and then we have diagnostic port, we are running diagnostic applications.

And just see here, that the signals generated by the external environment are sensed by the sensor and then the system processes this and then the actuators are driven and then this again controls the environment. So, you can think of this part, as a loop a feedback loop, the environment is getting controlled; each time events are sensed corrective actions are taken through the actuator and again the change of that correct, correct evacuation is sensed. Now, look at here, that there is an electromechanical backup and safety. If the system fails wherever possible, see we will see many applications, where such a electromechanical backup safety would not be possible, but many systems becomes possible and we have this to takeover, in case this system fails.

(Refer Slide Time: 11:14)



So, this is one basic diagram, we will keep in mind as we proceed. Now, before we get into the nitty-gritty of a real time operating system, the different components of it features and so on, let us just have a rough idea.

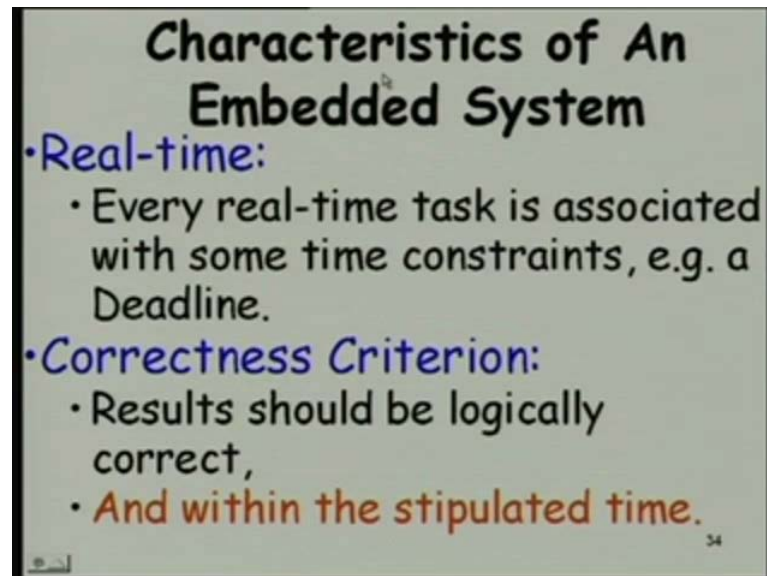
We, we had, so far said that, there are external inputs and then there are timing constraints on this, the behavior that will be determined based on the input and if the behavior at the response is too long, then we will say that the system fails, but using a general purpose operating system how do we ensure that, because there might with several types of events requiring different times by which they might be completing.

So, the general purpose operating system, we will also examine this issue much more depth as we proceed, but these cannot be used in these kinds of applications. So, we will have this real time operating systems, whose primary aim would be to help tasks meet the deadlines; how they will meet the deadlines, how deadlines are specified, all those details we will look at later, but just to tell in one sentence, these task deadlines they are met through task scheduling, proper task scheduling.

And therefore, in this real time operating system area, task scheduling is one of the most important task; you look at any book or whatever any discussion on real time system, you will see that, it revolves around the task scheduler, that is the crux of the thing. So,

we will spend some time on looking at the features of different schedulers, where they are used, what is their short coming, how to select a scheduler for an application.

(Refer Slide Time: 13:30)

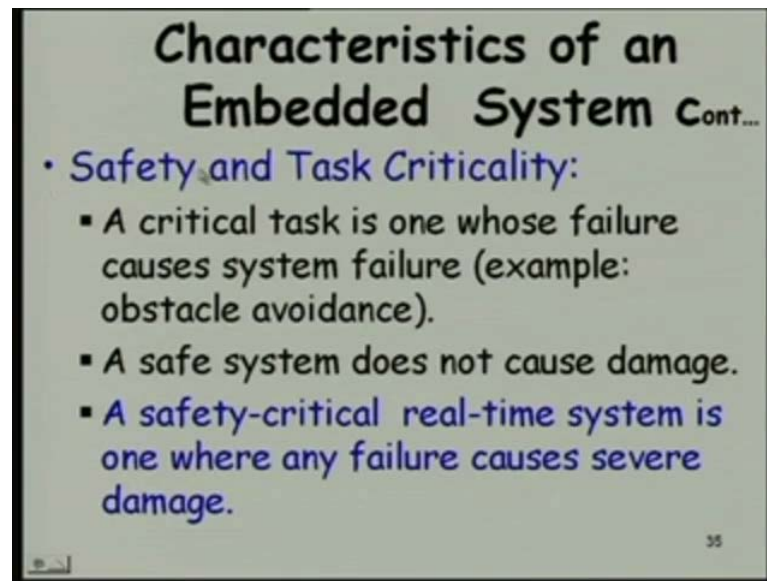


Because operating systems, might give you facility to configure different types of scheduler in your system. So, we will, we will look at those issues. Now, let us look at some basic characteristics of an embedded system, we will keep this in mind as we proceed.

So, one thing which we have said quite, a number of time that in a real time system, at least some of the tasks or some of the behavior associated with some time constraint, one example of a time constraint is a deadline constraint has to finish by sometime; another type of constraint, may be start the task, after sometime, after elapse of sometime the task will be started. We will see different types of constraints that we will encounter in a real time system.

Now, another characteristic here, is that, just having the result correct is not enough, suppose the corrective action for a chemical plant, when the concentration or the pressure or the temperature changes, the corrective action that needs to be taken maybe correct, but unless it is done at the right time, that one be correct, it will be a failure still. So, we will consider, it has a failure; now, let us look at the further characteristics.

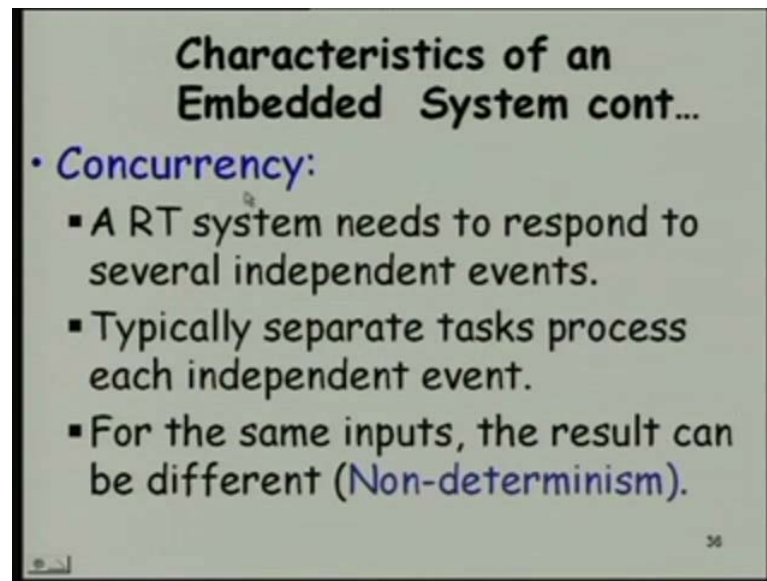
(Refer Slide Time: 14:50)



Another one important characteristic is safety and task criticality; we will call it has to be critical, when if the task fails then the system would fail. So, any failure of this task, of a critical task would not be accept it. Just to think of an example, let us consider a Robo and its moving and encounters and obstacle and suppose the obstacle avoidance task is a bit late, does not meet the deadline, then there will be a collision of the Robo and we will say that, see the system failed could not detect the obstacle and take correct action in time; so, that is a critical task.

And the other concept here, other than the criticality critical task, if there is a failure, it will fail the system will fail. And safe, let us look at concept of safety; a safe system is one, when it fails does not cause damage; if you have a safe system, we will be comfortable that there will be nothing can be really damaging out of it. So, these systems have to be safe, because damage on the tolerated (()), because this damage can be very costly, are used in life saving applications and critical applications, like a nuclear power plant. So, we will see, how this will be ensured later, but we are just looking at a characteristic that these systems criticality is a characteristic, some of the tasks will be critical safety requirement and these are called a safety critical; we will elaborate this issue a bit later.

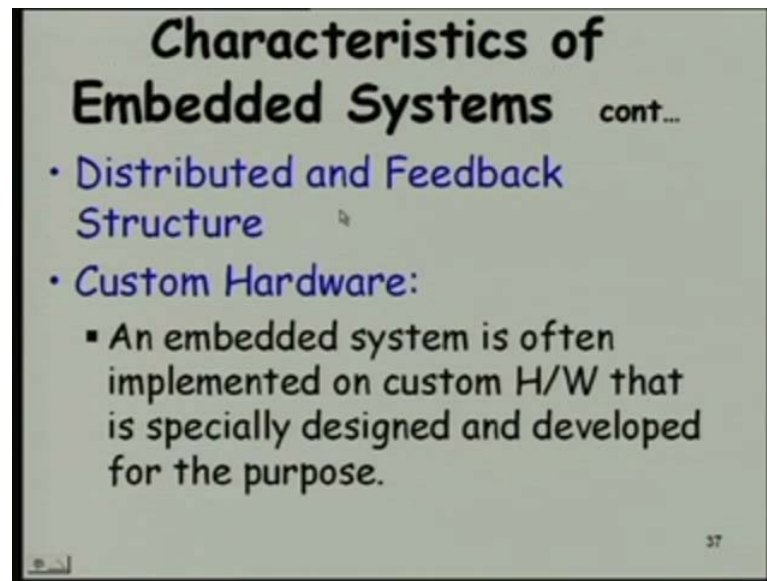
(Refer Slide Time: 17:13)



We will call these systems to be safety critical, some of these systems, where if there is any failure, there will be severe damage that will be caused. Now, there is another characteristic here, important characteristic through almost universally across all these systems, that these are concurrent; in the sense, that multiple tasks will be active, at the same time all the systems, we will see later, multiple tasks will be there and they will be active, but why do we have this requirement, why is it, that there will be multiple tasks can not one task be enough. To answer this question, we need to see how this operate; actually, this respond to events and the events can be many, like we are saying that, there may be pressure sensor, temperature sensor, may be chemical concentration sensor or may be in a (()) system, might have to sense the altitude, might have to sense the acceleration speed all those things.

So, multiple events need to be sensed at different intervals, because of the characteristics of the sensors or the characteristics of the events themselves and then, this has to be processed separately handled and then we will, we will, will get more idea as we proceed. Now, one consequence of these, is that, even that the input values are the same depending on when they occurred, the output may be different. Any concurrent system exhibits non determinism or the output will be different based on how this events are sequenced and in real time system, how these are spaced out in time domain. So, this also we will keep in mind, as we proceed important characteristic, I hope it is, is not it.

(Refer Slide Time: 19:13)

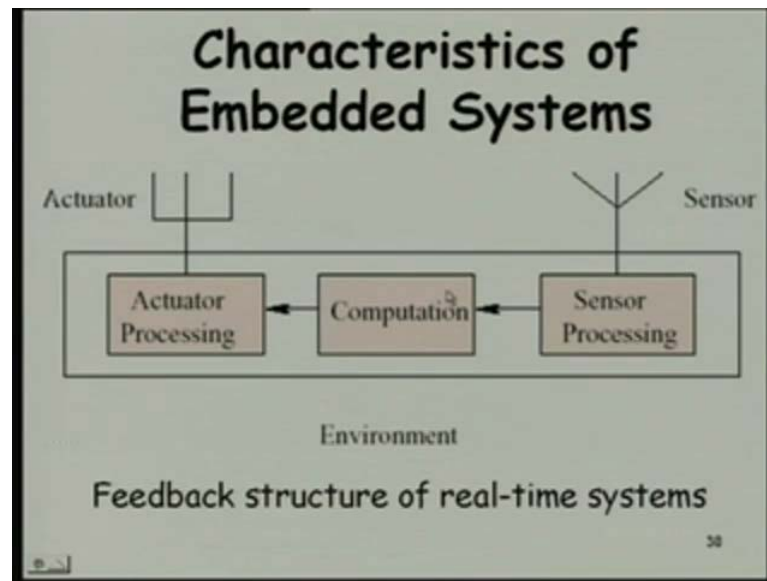


And as you are saying that this have feedback structure, because the environment is sensed, the events are processed, actions are taken and the response to that action is again sensed, but these are also distributed in nature, we will see majority of that large number of data distributed why is that, because events can occur anywhere.

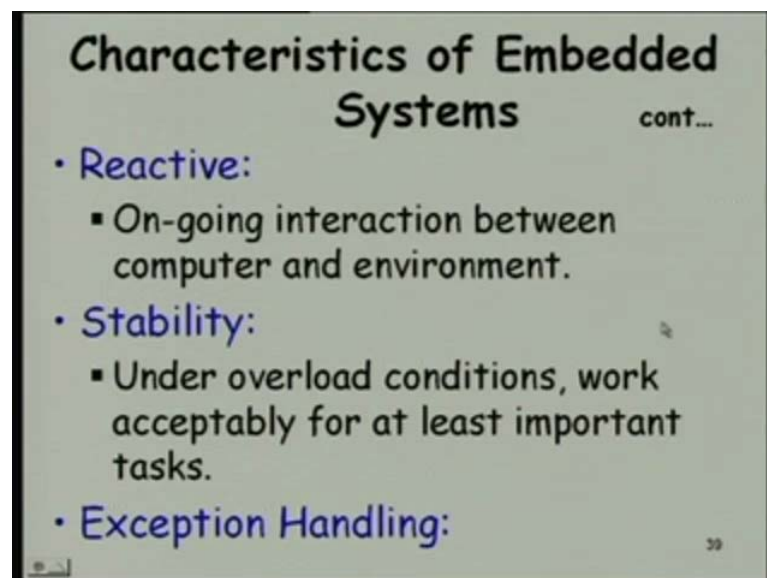
And we need to process those events which have occurred in a over a location, possibly at the same location and store at the same location, like we are discussing about the scada other applications. Another requirement is a custom hardware, for these applications do not need the general purpose processor or may be general process purpose sensor or actuator or you might have to design a specific hardware for many of these devices; for example, if we have a coffee vending machine, if we use a general processor, it will not fit into this, because it will be extremely expensive is not it; need a small processor there.

So, we have different kinds of embedded processors, for example, arm processors are important category of embedded processors, but we will see that the hardware, we do not use a general purpose board, general purpose hardware many times just for cost considerations size, considerations specific characteristics; we need only specific part of the processors. So, why do we invest on a full processor? So, these are some of the reasons, why many of these are based on custom hardware.

(Refer Slide Time: 21:06)



(Refer Slide Time: 21:57)



This we had seen in some form that the sensors generate signals and these signals are conditioned and read by the processor and computation occurs and then the actuator signals are generated to drive the actuators and there is a feedback nature. So, this is a much more simpler diagram, then what we discussed earlier, but let us keep this in mind as we proceed, that there will be signal sensed computation occurring, on this actuator will act on the environment and the change in the environment will again be sensed. A

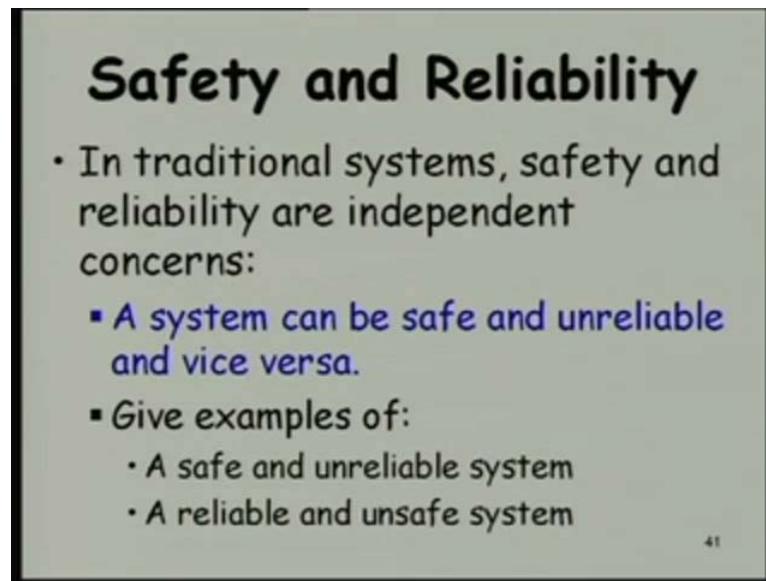
few more characteristics, we you might also find these terms, as you read literature on real time system, one is called as a reactive system.

Actually, a many of the applications that we use for example, you are using a word package, you use it for some time and close it down, that is not a reactive system, but a reactive system is one, where it just continues to occur over a large period of time. So, there is a ongoing interaction between the computer and the environment over a long period of time on attended by anybody, just keeps on working for months or years together. So, this reaction between the computer and the environment over a long period of time is called as a reactive; this term is used in the literature, you read a book or a paper, you will come across this term. Another term that we will also use as we proceed is stability.

Now, let us get this term clarified, that sometimes events, see events when they occur do not have much control, we have control on the system, but environment will behave on its own way and what if several events occur at the same time, let us, **a**, all these are being sensed, temperature, pressure etcetera and also there is a fire situation which is monitored. So, several events will get generated at the same time and can the system process, all this at the same time.

So, that is an over load situation and if the system fails under this situation, then will say that system is unstable under heavy load. So, this will be an important consideration, as we look at the operating systems and so on. We will talk about whether the system is stable under over load situations, is that, the stability under over loads situation, let us look further. Another one will be exception handling, in case on exception events occur how this will be handled, we will also discuss in the context of the operating system exception handling.

(Refer Slide Time: 24:28)



Safety and Reliability

- In traditional systems, safety and reliability are independent concerns:
 - A system can be safe and unreliable and vice versa.
 - Give examples of:
 - A safe and unreliable system
 - A reliable and unsafe system

41

Now, let us elaborate this concept of safety and reliability, because important issue in a real time system. In a traditional systems, if we have this concepts also safety and reliability in traditional systems and there we have these concepts of a safe system and reliable or independent, for example, we will have a safe system which is not reliable and we can have a reliable system but not safe.

We had already seen this actually, a safe system even if it fails, no damage will result, on the other hand, a reliable system is one, which operates for long periods of time without any failure being encountered, then we will call it as a reliable system. In traditional systems, safety, we can know increase safety or we can increase reliability without affecting each other and we have, a, we can also have systems which are safe, but unreliable, unreliable and unsafe reliable and unsafe and so on. In the real life situation, can you give an example of a safe system, but which is unreliable, can anybody give an example of a safe system, yes.

Mobile phone. Mobile phone, mobile phone is reliable is not it, I do not know what kind of mobile phone you have, but normally it is a quite reliable is not it, (()), for long period of time, but much failure reliable mobile phone is suppose to be reliable. Anybody has any example of a real life situation need not be a computer anything in real life, can somebody tell a system or an artifact or whatever which is... (()). Weather forecast system, is it unreliable, I mean it works. (()). Very unreliable. (()).

I mean as long as the letters you send, if they are lost you do not mind, then it is, you can say that it is a safe system, because even if letters are lost, it does not really hurt you or may be email system, where emails may get lost they are getting lost. So, this is safe, but again unreliable, email, know you have seen it does not reach, but whatever the other example.

We seen a safe and unreliable system or the another example can be that, you have developed a word processing software crosses very frequently, like earlier times the word processors use to cross, now they are more stable. And before it crosses, it just saves the data. So, you do not lose much its only irritation, that you need to start again and again, but you do not much data, that is another example of a safe and unreliable system, what about an unsafe and reliable system can somebody give an example? (()).

Unsafe, why is it unsafe? (()). It is a cardiac monitor example you gave, so, why is it unsafe? Because surgery is going on; ok. Then it will safe there (()) patient (()). Ok.

Patient can die (()). Yes, but there can be several examples, you can just say that a gun, the very reliable, does not fail normally, but if it fails or let us say, the, your firing it does not really, I mean, it explodes there is a problem, it happens you know. So, it is a safe and it is unsafe, but reliable system, now let us proceed further.

(Refer Slide Time: 29:14)

Safety and Reliability
cont....

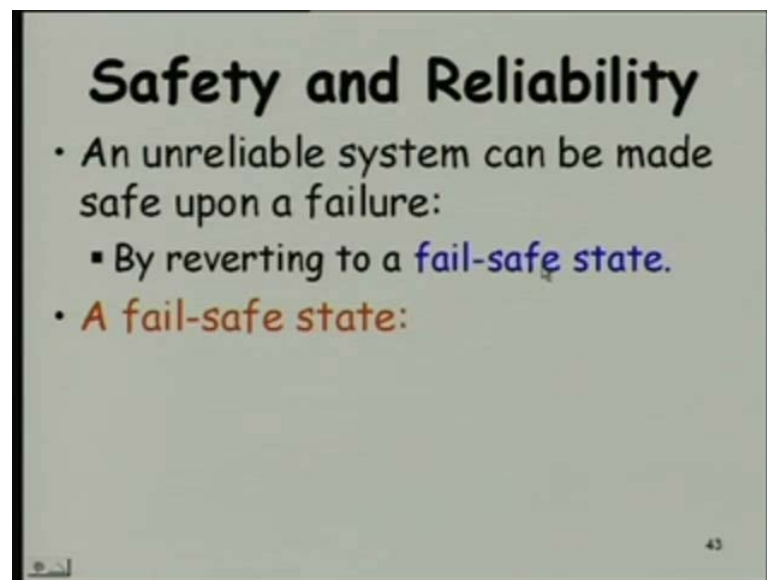
- Interrelated in safety-critical system.
 - A safety critical system is one for which any failure of the system would result in severe damage.
 - Safety can be ensured only through increased reliability.

42

In traditional systems, safety and reliability are independent issue, we had seen (()) system, safe, unreliable, unreliable unsafe and so on, but in a real time application, which are many of them are safety critical and in these kind of application, any failure at the system would result in severe damage. So, we cannot, let the system fail and also it has to be, you know, should not cause damage, just see here (()), the system fail and cannot cause damage, even if it fails. So, these are called a safety critical.

Now, let us see the implication of this; the implication is that safety and reliability are no more independent issues, they are dependent on each other; it has to be reliable and also at the same time safe; so, how do we get that.

(Refer Slide Time: 30:40)



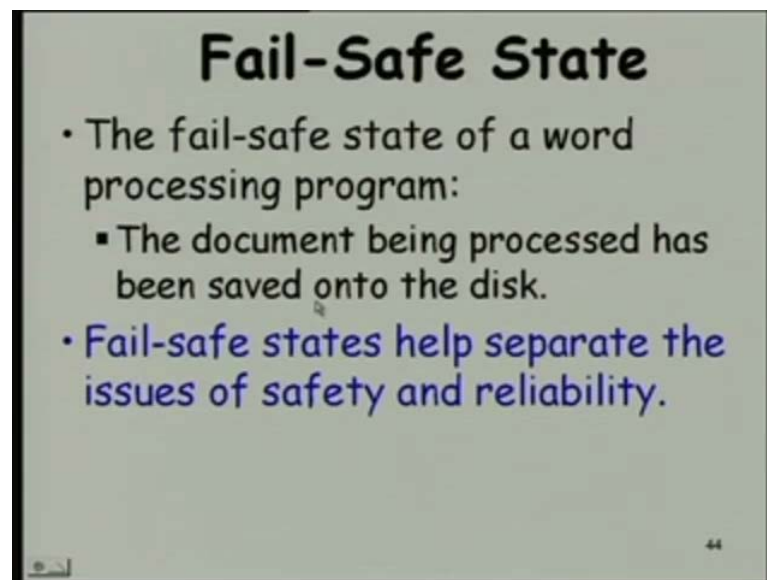
We have to have increased, reliability should never fail. So, one possibility it should never fail in a traditional system, you can make a unreliable system to be safe by reverting to a fail-safe state; just let us take that example of a word processor, your valuable data and it was unreliable, but before it crossed or something, it just save the data. So, there is a safe state for that, if the safe, it fails in the safe state; all data has been saved then you do not lose much.

So, it is unreliable system, but we have made it safe, just by transiting to a safe state during failure or let us. So, a safe state is one, where no damage will result, if the system fails in the fail- safe state. So, if a fail-safe sate exists, we can convert a unsafe,

unreliable system to a safe system; we will just transit to the safe state, this is an example here see the a traffic light controller in road intersection, even if it fails, it leaves the lights orange and blinking this is a safe state here; suppose, it failed with lights, all green at the intersection, it will become unsafe there will be accidents people will rush in.

If it fails in all red that will be unsafe also is not it; traffic will become a standstill, but **if it**, there is a safe state like orange and blinking, people know that it is not working they proceed with their discussion. So, we will also use these terms a fail-safe state, safety criticality, safety and reliability all these terms we will use it, I hope these terms are clear **ok**.

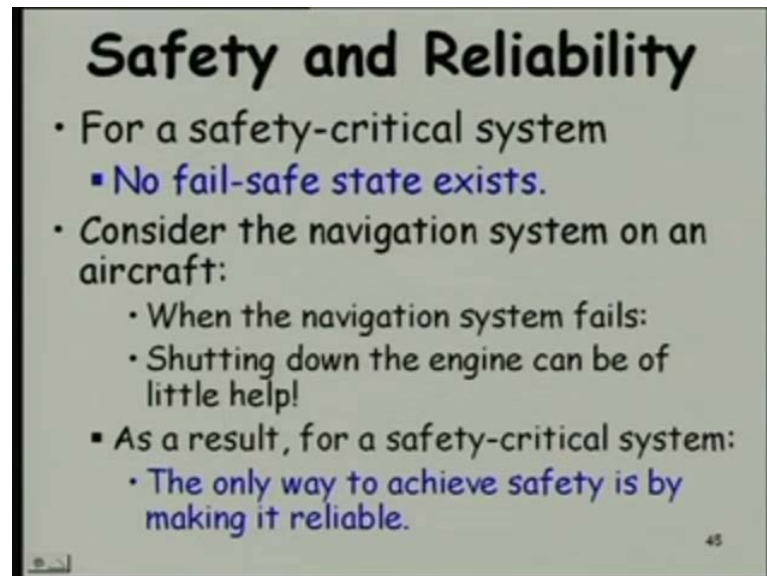
(Refer Slide Time: 32:55)



We, so, we had already seen that for a unreliable document processing system or a word processing system the fail-safe state is saving the document in the disk. So, a fail-safe state actually helps to separate the safety and reliability issues, because we can have an unreliable system and we can make it safe, but unfortunately in real time systems, safety-critical systems, we will see that there are no fail-safe sates, otherwise we would not have much problem; even if it fails, we will, **we will** change it to or we will, **we will** make the system fail in a fail-safe sate; unfortunately, for this system, as we will see shortly fail-safe systems, do not exist in a safety-critical system **ok**.

So, an unreliable system can be converted to a safe system, unreliable and non safe system can be converted to a safe system, just by making it fail in a fail-safe state.

(Refer Slide Time: 34:23)



Safety and Reliability

- For a safety-critical system
 - No fail-safe state exists.
- Consider the navigation system on an aircraft:
 - When the navigation system fails:
 - Shutting down the engine can be of little help!
 - As a result, for a safety-critical system:
 - The only way to achieve safety is by making it reliable.

45

Now, as we will see the safety-critical systems, they exist no fail-safe state; if it was a system, where fail-safe state exist and even if it is a critical system could possibly have, we would have change the state to a fail-safe state and no damage will result. Now, let us take an example, let us say there is a autopilot system and the plane is flying and suddenly the system fails, then what will be the fail-safe state would setting down the engine help, would, no, it will still fall is not it.

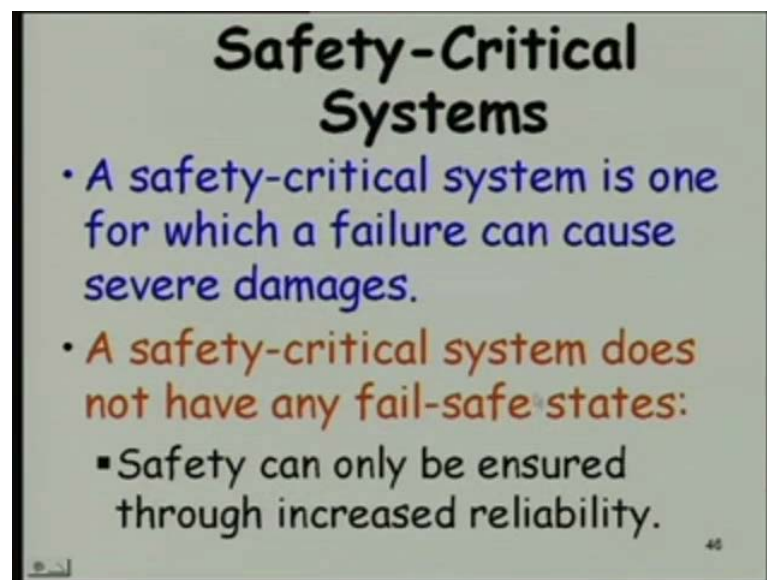
Sir, it can be taken over by (()), Ok, but if the manual pilot exists and he can control, but many of these autopilot, you know, they, they just fly by themselves and sometimes, it is not possible for the auto manual pilot to take over a plane; for example, let us see this plane, supersonic commercial plane, have you heard of this concurred, anybody heard of concurred flight.

So, it was a supersonic flight operated between I think London and Newyork, the normal planes will take something like eight hours and this a supersonic commercial plane; see we have seen supersonic planes are the jet fighter planes and military application, where there is a commercial plane and use to operate 4 hours, it use to take.

And then just crashed and they have just said that the autopilot system failed but nothing could be done it fly at that speed and a large aircraft, even if a pilot is there, he will be of little, you know, control he can control such a large plane, there are kind of response, time etcetera, will be required at that speed will be difficult is not it.

So, for such systems, a safety-critical system is not even asking for the manual pilot to take over. So, for a safety-critical system, the only way we can make it safe, by making it extremely reliable for a million flying hours, one failure, million flying hours, that is, the requirement of the federal aviation agency extremely reliable, let us proceed further.

(Refer Slide Time: 37:14)



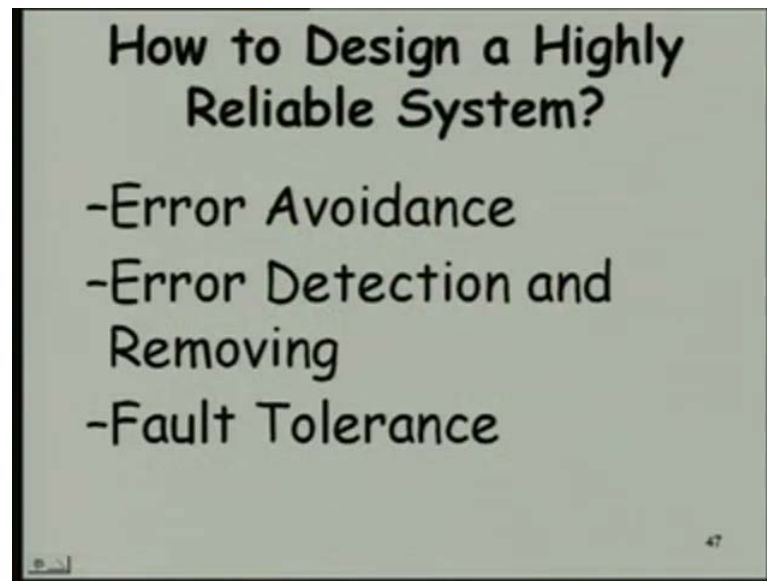
Safety-Critical Systems

- A safety-critical system is one for which a failure can cause severe damages.
- A safety-critical system does not have any fail-safe states:
 - Safety can only be ensured through increased reliability.

46

So, just to summarize what we discussed, a safety-critical system is one, where any failure can cause severe damage and no fail-safe state exists in the safety-critical systems. So, cannot even revert to a fail-safe state.

(Refer Slide Time: 37:46)



So, the reliability will be increased only through, in the safety will be ensured only by increasing the reliability making it extremely reliable, but the question is that, if you want to develop a highly reliable system, how do you do that? Let us not worry about hardware, because that is not our domain; let us see, as far as software is concerned how do develop extremely reliable software? Yes, anybody would like to answer, how do you develop extremely reliable software? (()). Testing. (()).

Is this testing, yes, we have to test them, but we will know the testing does not really reveal all problems in a system. Even after a thoroughly tested, you cannot guaranty that there are no problems. So, what do you do, testing, yes you have to test it very thoroughly correct, but that would not be really solve the problem.

Sir, we can use the development methodology like some model (()). Ok. (()).

So, to reduce the scope of error, you can use some methodology, which will reduce the chances of errors, but still there will be some errors remain you cannot guaranty that, **it is**, there are no errors in the system; a large systems at the present moment, it is impossible to guaranty that there will be no failure. So, in case of a failure, we will have to provide for fault-tolerance; even if the system fails, there should be some mechanism in the system, which will in the form of a redundancy possibly, that it will take over.

Even if the suppose to a failed, another thing will take over and it will not really fail; so, let us see these approaches.

Is one is error avoidance, as he was saying there, that we can use some systematic methodology, where we reduce the chances of error, we avoid errors; the other is testing detect errors and then we remove the errors, test it thoroughly and still there will be chances of problems, because we cannot really test with all existed values; there are too many of that to able to test in a reasonable time. So, we will have to have fault-tolerance features in this kind of safety-critical systems **even if... (())**

No, fail-safe is a state, let me just repeat his question says that, the terms fail-safe and fault-tolerance appear to be confusing that is the question. See fail-safe is the system fails, but while failing it just fails in a state where damage will not occur. But in fault-tolerance we are saying that even if there is a problem somewhere, we are masking it, by some means and how we are masking is through redundancy; for example, it is very easy to consider the case of a hardware a processor man malfunctioned or a sensor malfunction, we have another redundant sensor. We will just reconfigure that the processor takes over and even, the one processor has failed or a sensor has failed, we will not really notice that for the operation; it will make little impact, even if the processor that was working that really failed, because another extra processor has taken over. So, that is an example of a fault-tolerance is that, **ok** let us proceed.

(Refer Slide Time: 41:51)

Fault Tolerance in RT System

- The essential idea:
 - **Provide redundancy**
- **Hardware Fault-Tolerance:**
 - Masks the effects of a hardware fault.
- **Software Fault-Tolerance:**
 - Masks the effects of a program fault.

40

Fault-tolerance in hardware is very intuitive; if you talk to persons working in the hardware area, they say that yes, the hardware that we develop are fault tolerant and there are standard technique called as triple modular redundancy built in self test.

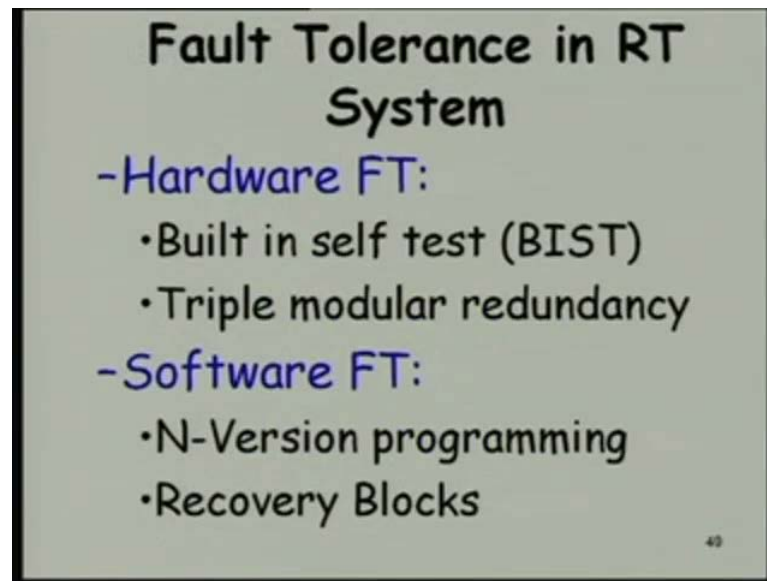
So, you just find errors and then you invoke another one or you do a voting, you have three redundant pieces, all working at the same time and then, if one of them fails, the other two the results will differ and you know that there is a failure and then you remove that one which failed and use that majority result, standard technique triple modular redundancy is the standard hardware fault-tolerance techniques, but we will see that, in case of software which is the disk, our current discussion say very tricky, so let us see what are the approaches to fault-tolerance in software and what are their problem, because we have to keep this in mind as we proceed. See I have, we had seen that the idea behind any fault tolerant system to have some form of redundancy maybe redundancy of components, redundancy of programs or maybe redundancy of some activity.

In hardware, in hardware fault-tolerance, we can easily mask the fault of the hardware as I said with triple modular redundancy, but the fault maybe due to software and that is more likely, because hardware nowadays, so reliable, they hardly fail, but even if there are chances of failure; we can provide fault-tolerance in hardware, very easily, but if the fault is due to a software; even if you have hardware redundancy all those nothing will work, it will still fail. So, how do you provide software fault-tolerance, there is bug in the program, how do you really provide a fault-tolerance feature in software? Anybody would like to answer this question; I am sorry. (()).

Not exception handling actually, see the let us say the program has gone into an infinite loop, so what do you do; exception handling is not really fault-tolerance. (()). Anybody else would like to answer. (()). No, but the software running on the same is not it. Even in a distributed system, the same software will run there; so, there is a bug and it will fail again for the same reason. Sir, can we have a monitor program (()).

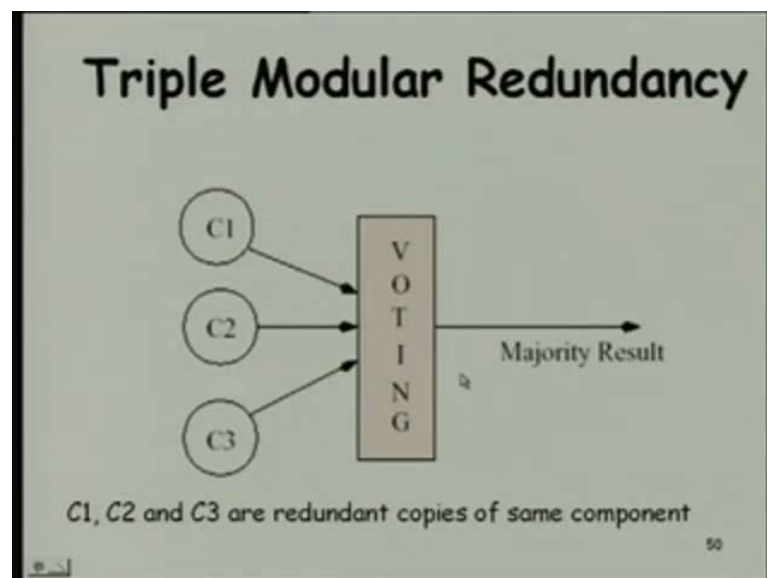
No, how will you overcomes, see again if you run that software, again that problem will come up and you run another copy that problem will be there like, you are saying in a distributed, we have another copy which will run, but again that problem will come up ok.

(Refer Slide Time: 45:38)



So, there are two standard approaches to software fault-tolerance; now, let us look at these approaches. One is called as N-version programming and the other is called as recovery block; two very popular software fault-tolerance approaches used also in practical situations, but we will see that not effective.

(Refer Slide Time: 46:12)

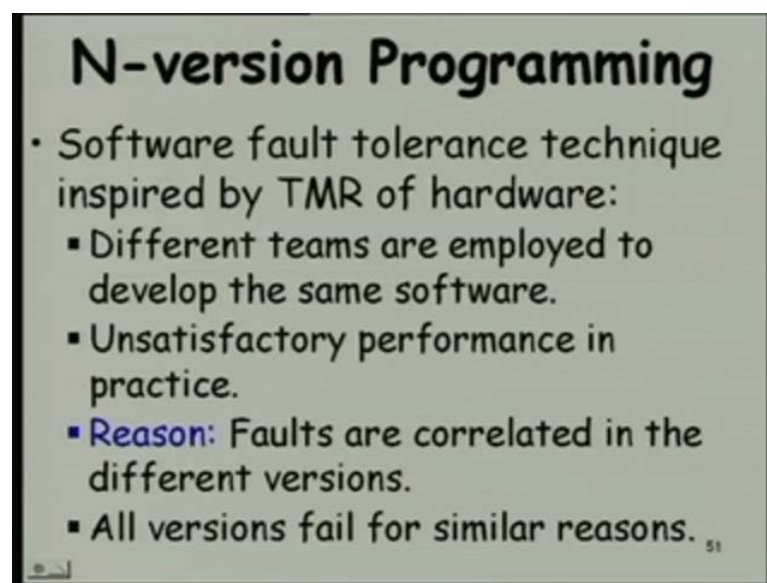


Software fault-tolerance still becomes, still remains a very challenging issue, no really good technique like hardware, fault-tolerance exist. Now, let us look at this N-version

programming, actually the N-version programming is based on the hardware redundancy technique of triple modular redundancy or T M R.

In a T M R, as you are saying that there are redundant components, three of them, we have even though one could have served ,we have three of them repeated and we take the voting of the results and use the majority result; now, this is extended to the software domain, how do extend this to the software domain? In software domain, we create different versions of the software; we have independent teams that are working.

(Refer Slide Time: 46:47)



N-version Programming

- Software fault tolerance technique inspired by TMR of hardware:
 - Different teams are employed to develop the same software.
 - Unsatisfactory performance in practice.
 - **Reason:** Faults are correlated in the different versions.
 - All versions fail for similar reasons.

51

So, the same problem, three different teams they are developing three versions of the software and hoping that, they will not commit identical mistakes in their code or they are algorithm; they might use different algorithms also, because they are working independently hoping that they are different versions of the software will be different. We can make them work at the same time and then take the voting, but again is not a very effective technique, unsatisfactory performance why is that. Actually, even if we have independent versions, independent teams working, but when you actually run them, you find that the faults are **the under** the conditions, the fail are identical or the faults are similar, why is that, because teams were different, different types of people working at different places, they do not even talk to each other. **(())**.

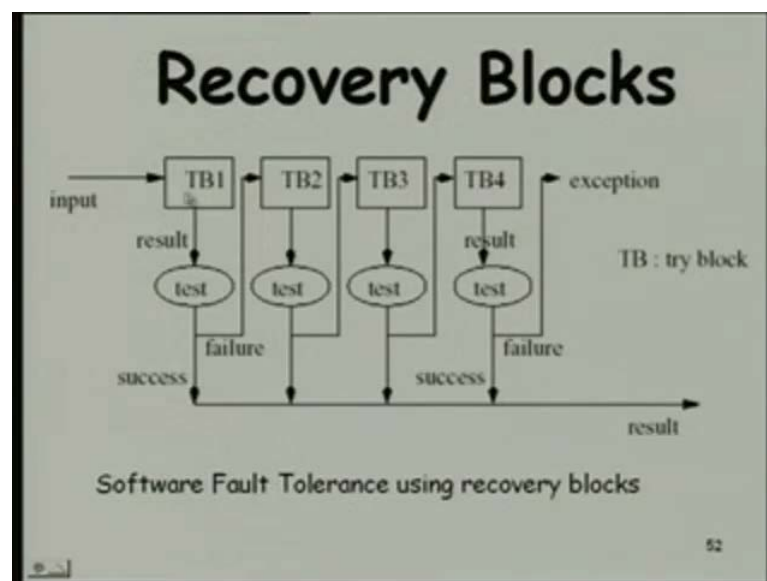
No, input is the same, but like let us says programmer committed a mistake and there we have a different programmer, we might not commit the same mistake, but we are finding that they are failing in similar reasons, why is that. (()), but they can be perfect ((design level which are not.

No, designed also differently, the different teams they do their design requirements; they do their coding testing independently. They are following the ((). Need not be, they might use different program. ((). Methodology, ((relating to the complexity of the (().

Maybe, that is the answer, see the thing is that, even if the programmers are at different places, they are working without talking to each other, but the part of the program where one team finds difficulty and likely to do mistakes is the same for an any other team, they find that the same modules are difficult to code and their errors remain.

So, maybe for that reason, the N-version programming is not very effective, I mean, it might work for some simple programming errors, which one team committed another team did not commit, but for majority of the errors, it does not work actually, because these are identical errors in the different versions of the program.

(Refer Slide Time: 49:44)

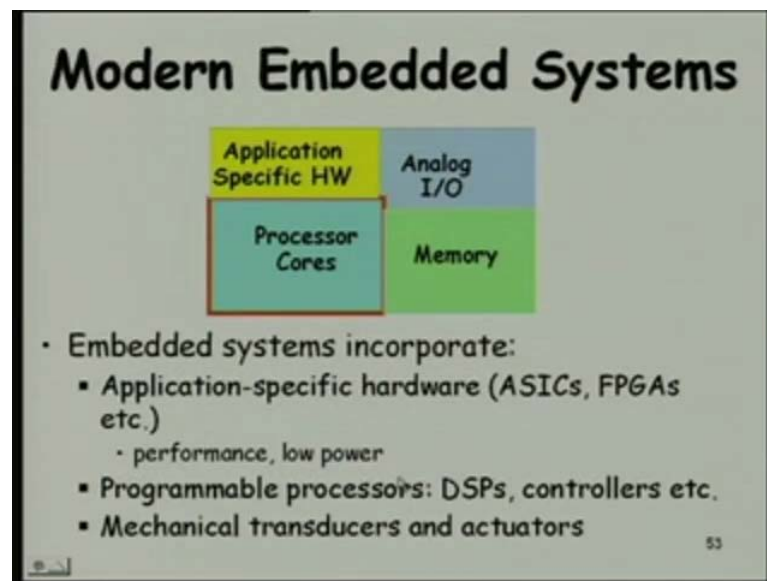


Now, let us look at the other technique the recovery block technique. Here, we have different a task is divided into try blocks; so, one task is divided into four try blocks, I

will let me just correct myself, these are not the task is not really, these are not really T B 1, T B 2 which are the try blocks, is a not really diffusion of a task into these four blocks, but task is divide into four or may be more blocks, which we do not see here; see these try blocks that we see here are same try block, I mean, same type of programming same part of the task, they are doing, is a redundant copies again, but thing is that, only one operates, try block one operates and then the result is tested; if it is a failure, try to use another try block, again the result is **tested**.

So, here just see that, they are not really redundant copies or different versions, but these are different try blocks that are run one after other and in a real time **situation**, by the time the try block one is completed, it takes some time; so, you have still less time remaining to try the other try block, So, possibly you can have a simpler algorithm here or maybe you can sacrifice accuracy or something.

(Refer Slide Time: 52:06)



And these use deliberately different algorithms. So, this is also another way to provide software fault-tolerance, but again the same problem, correlation of the errors and is not very effective. **(())**. **(())**. **Yes**. **Only one (())**.

No, see here the try block one is the actual module and we test it; if it is a failure, we try a simpler algorithm which is using a different approach. **Sir...** Different algorithm,

deliberately different each try block is deliberately a different algorithm. (()). Sorry. (())
N-version (())

In N-version, we do not say that they are different algorithms, we say that independent teams, they might use the same algorithm; it might be a standard, they just do not talk to each other. So, may be if it is a standard, let us say a quick sort algorithm is used by one team, the, another team might also use the same algorithm, because that is the most effective at that time. But here you are saying that, they do not use the same approach, deliberately use different approach and of course, this have to be progressively simpler algorithms with sacrificing accuracy, because time is becoming smaller and smaller, the time remaining to complete is smaller and smaller.

And once we have these try blocks, if there is a... We can also restart it, that is, one thing that we need to understand that, see the input has been processed here, but we can, the, try block two can be restarted, because we have to recover we can go to a previous state a recovery state and from that state, we will try the try block. So, each try block, after completion, we will have a recovery state, where we just to overall the results; we can always go back to that state and then restart that is the idea. (()) get the input other than the (()).

We have, what we have showed is T B 2, in case of a failure is invoked, I think there is a accuracy, inaccuracy in the sense that, it will revert back, it will recover, I mean, it will go back to the previous state. So, we will just stop here and we will continue in the next lecture.