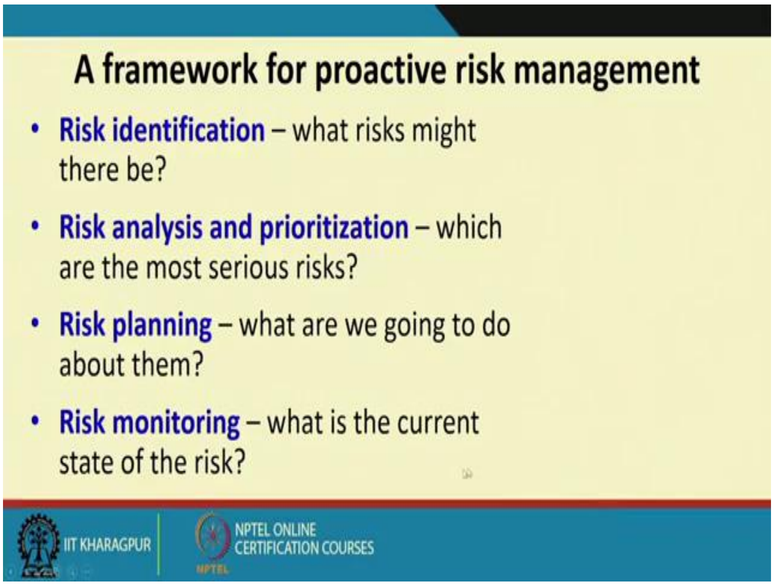
Software Project Management
Prof. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 35
Risk Management (Contd.) and Introduction to Software Quality

Welcome to this lecture, in the last lecture we had looked at some basic concepts on Risk Management we had discussed what is a risk different types of risk and a framework for risk management. Today we will discuss further on risk management and after that we will start discussing about quality management; let us get started.

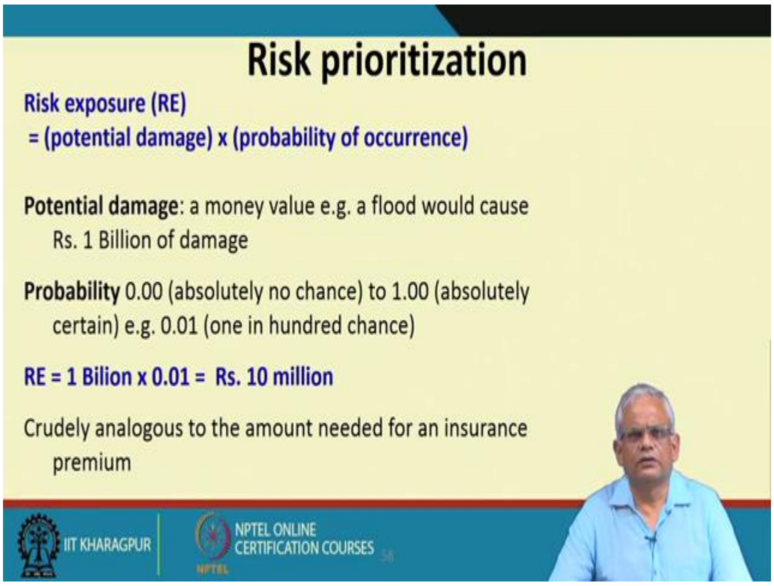
(Refer Slide Time: 00:55)



In the last lecture, we had said that there are two main ways for risk management; one is a proactive approach where we identify and make plans for a risk. In a reactive approach, we do not make any specific plans; possibly the risks are hard to anticipate we handle the risks as they appear during the project. In the proactive risk management, the first task is risk identification. We list down all the risks we had discussed some guidelines about how to go about identifying the risks and once we have a list of risks, we do an analysis of risks.

The analysis is in the form of what is the probability of the risk becoming true and what will be the cost implication of that risk. And based on that we can prioritize the risk the one which will be most damaging to the institute to the project that will be the most serious risk; we need to make elaborate plans for that. In the risk planning we will try to see that how the effect of the risk can be either avoided or minimized.

(Refer Slide Time: 02:55)

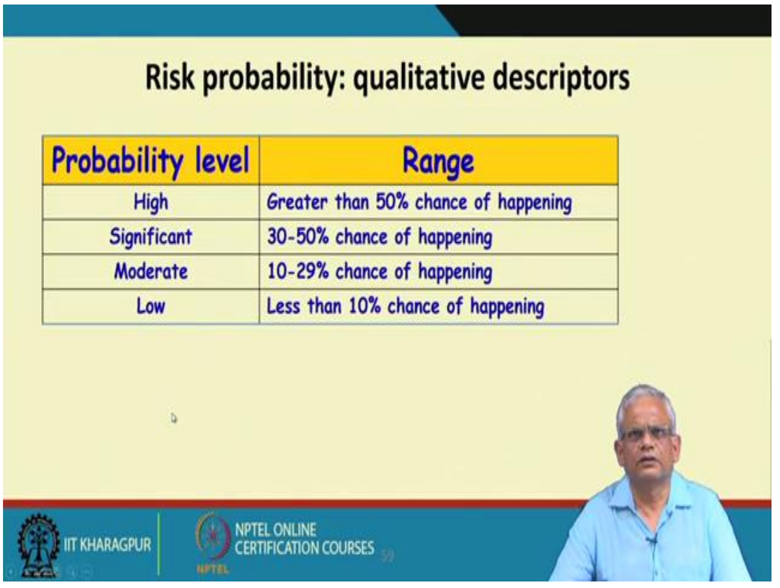


And finally, the risk monitoring where we see that what is the status of the risk. Once we have listed all the risks in a project to the project manager; we need to anticipate the risks. And once we have made the list of risks we need to analyze the risks; we mean analysis of the risk means that we qualitatively assign a probability of occurrence; it will be very hard to give a quantitative value for example, we cannot say that the occurrence of this risk is 66.5.

It will be hard and also we qualitatively identify or estimate the potential damage due to the risk. And based on this, we compute a Risk Exposure or RE for all the risks which is the multiplication of the potential damage and the probability of occurrence. The potential damage we assign a money value approximate money value for the risk.

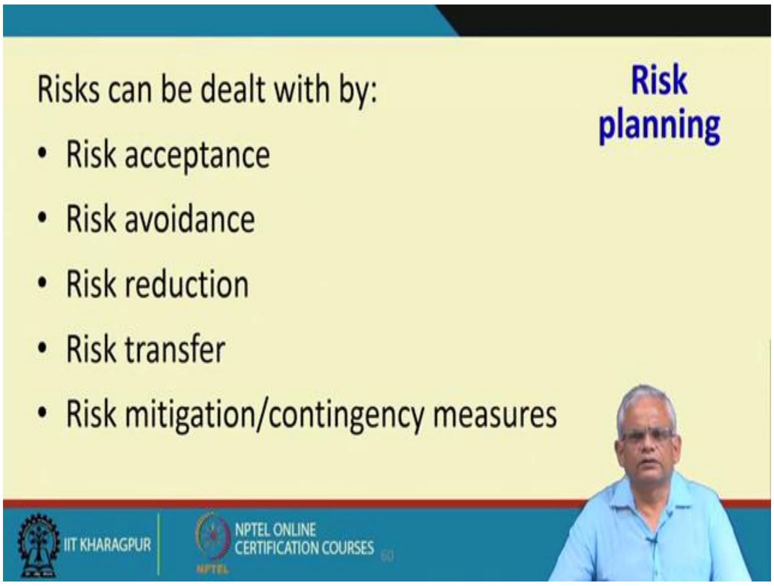
For example, a flood would cause 1 billion of damage and the probability of the flood occurring is 0.01; that is 1 in a 100 chance a rare chance. And for this two values the potential damage is 1 billion and the probability of the risk occurring is 0.01; we can compute the risk exposure for this case is 1 billion into 0.01 which is 10 million. We can think of this as the insurance premium; the insurance is basically computed something similar way that what is the probability of an accident occurring and what is the cost implications of the accident and that would give an indication of the insurance premium to be charged.

(Refer Slide Time: 05:41)



It is hard to give a quantitative value to the risks, often we can tell qualitatively that whether the chances are very high, the chances are significant or moderate and low. And based on this qualitative probability of a risk; we can give quantitative values. When we say that the risk is high it is something greater than 50 percent; we give a value greater than 50 percent based on the qualitative estimation of the risk probability. Significant is something like 30 to 50 percent, moderate is 10 to 30 percent and low is less than 10 percent.

(Refer Slide Time: 06:49)



And once we have prioritized risks based on the risk exposure computed for various risks, we need to do risk planning for the most damaging risks. The various ways that risks can be dealt with a risk acceptance, risk avoidance, risk reduction, risk transfer, risk mitigation and contingency measures. In risk acceptance, we determine that the things that we need to do; to handle the risk is more expensive than the risk itself.

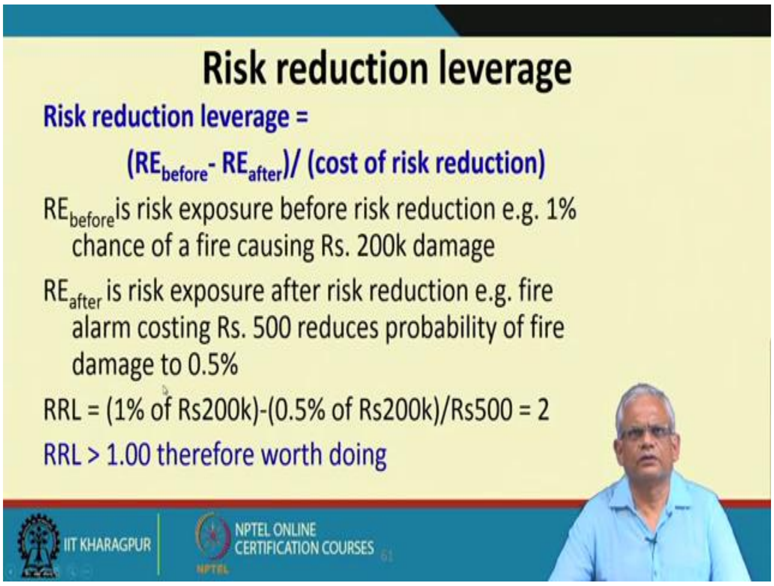
The risk acceptance is done for very low risk exposures; the damage will be very less and the probability is less; so this we do the risk acceptance. For these cases maybe the things that we need to do to handle manage the risk will be much more expensive than the risk itself; in that case it is better to have the risk acceptance. And even sometimes with risks with high risk exposure; it may so happen that we can do we cannot do much about that.

For example, that one of the key person leaving we cannot really employ a another person in the start the project that will be too expensive; it will be more damaging than the risk itself; so in this case we do the risk acceptance. Risk avoidance if it is possible to avoid the risk for example, if we find that one of the modules of the software is very difficult to develop and we are likely to have delay, bugs in this module; we may outsource it. So, here we have avoided the risk because we did not have capability and we gave it to some third party who have good capability of developing the module and we have avoided the risk.

Risk reduction in this case we make plans so that the impact of the risk will be less. For example, if we anticipate that some key persons might leave we might have good documentation; we might have review so that other members are aware of the work that is carried out by the key person. In this case, the new person who comes might refer to the documents already prepared and also discuss with others who have reviewed the work.

Risk transfer here we might buy insurance so that the risk is transferred to the insurance company we give premium but then the risk is now transferred to the insurance company. Risk mitigation and contingency measures we might prepare plans; if risk happens what contingency measures we can take.

(Refer Slide Time: 11:01)



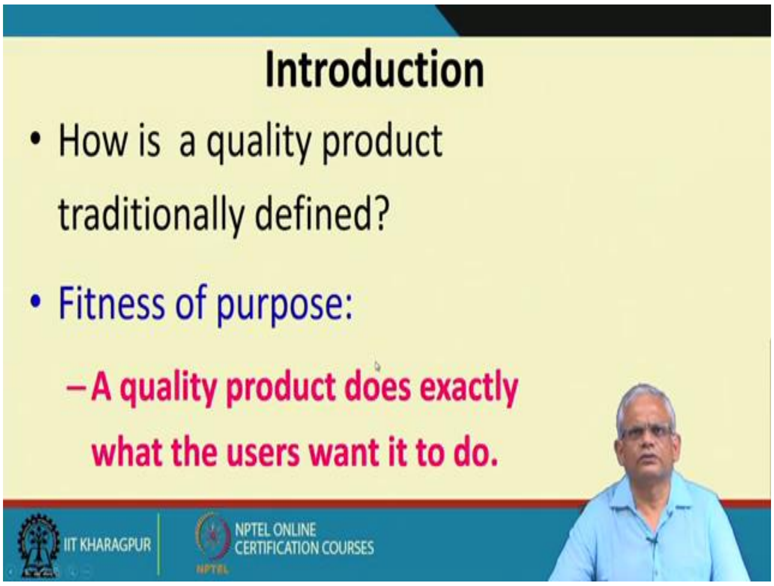
The risk reduction leverage, this is the effectiveness of a risk handling mechanism. The risk reduction leverage is a quantitative value for the effectiveness of a risk handling technique. Let us say we deploy a some risk handling technique and the risk handling technique; the cost of the risk handling technique and the reduction of the impact; we consider this here that the reduction in impact is; the risk exposure before deploying the risk handling technique minus the risk exposure after the risk handling technique is deployed.

So, this is the reduction in exposure after the risk handling technique is deployed and this is the cost of the risk handling technique. For example, that initially we had 1 percent chance of a fire causing 200 k damage.

So, the potential damage is 200000 and the probability of the risk occurring is 1 percent; so, the risk exposure before the risk handling technique is deployed is 1 percent of 200000. And now that we have installed a fire alarm which costs 500 rupees and now the probability of the damage due to a fire is 50 percent. So, 1 percent of 200 minus 0.5 percent of 200000 which is equal to 0.5 percent of 200000 which is equal to 100000; 1 percent of 200000 is 2000 minus 0.5 of 200000; so, it is equal to 100 sorry 1000 and 1000 by 500 is equal to 2.

And therefore, we benefit by deploying the risk reduction technique which is a fire alarm costs 500, but then the risk exposure reduces by 1000 and therefore, the risk reduction leverage is greater than 1. And whenever the risk reduction leverage is greater than 1, we say that this risk handling technique is worth doing it. With this discussion, we will conclude the risk management and we will discuss about Software Quality management.

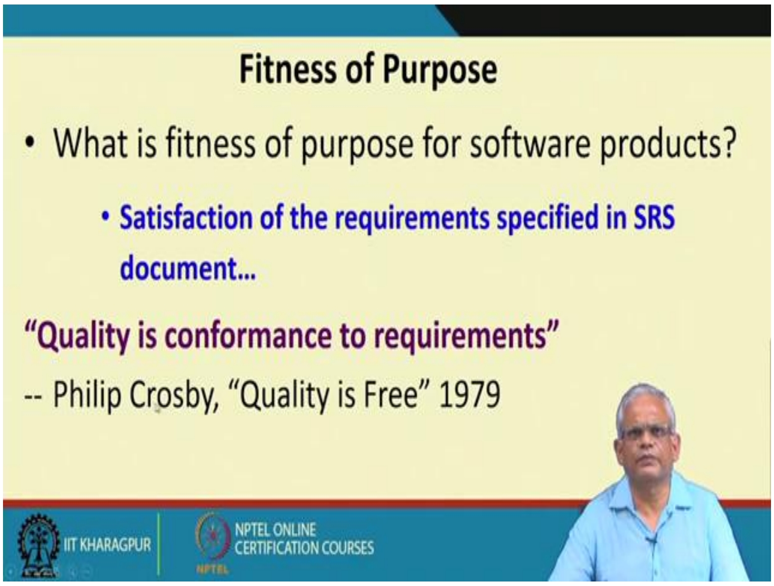
(Refer Slide Time: 14:39)



How do we define a quality product? Which one would you would you call as a quality product? Traditionally we say that a product is of quality; if it does what exactly the user wants it to do; this is the traditional definition of quality, it is called as fitness of purpose. For example, you want to buy a table fan and let us say you got one from the market.

And found that it runs smoothly and gives good wind; you say that it is a good quality product.

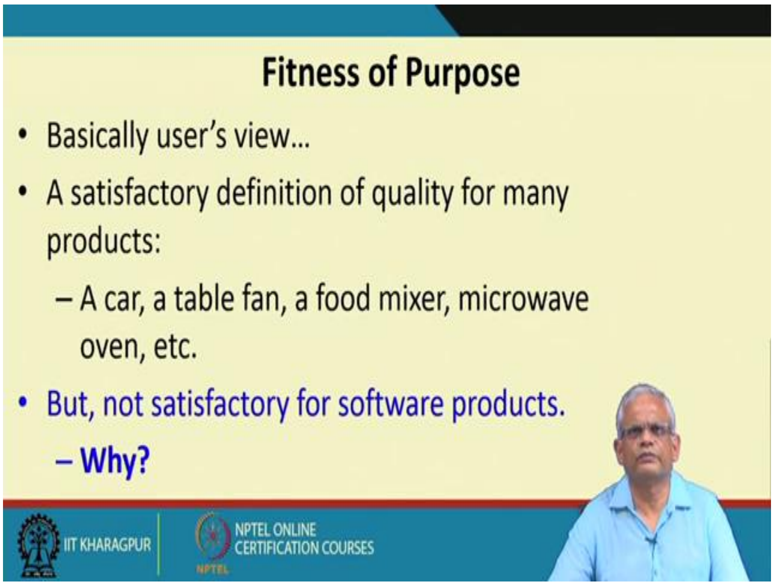
(Refer Slide Time: 15:45)



But for traditional items like a mixer, grinder, a table fan or a car, a bicycle the fitness of purpose is intuitively clear. But what is the fitness of purpose of a software product? When do you say that software product has fitness of purpose?

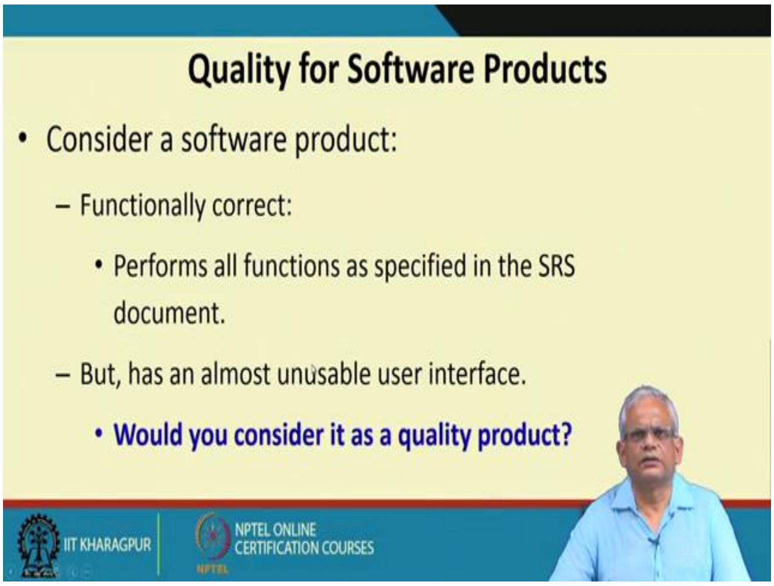
One way to define the fitness of purpose of a software product is to say that all the requirement specified in the SRS document have been satisfied. In other words, for a software product quality of the software product is essentially conformance to its requirements; every software is developed with some requirements in mind and as long as it meets all the requirements we might be tempted to say that it is a quality software. And its accepted by many including Philip Crosby in his "Quality is Free" book.

(Refer Slide Time: 17:05)



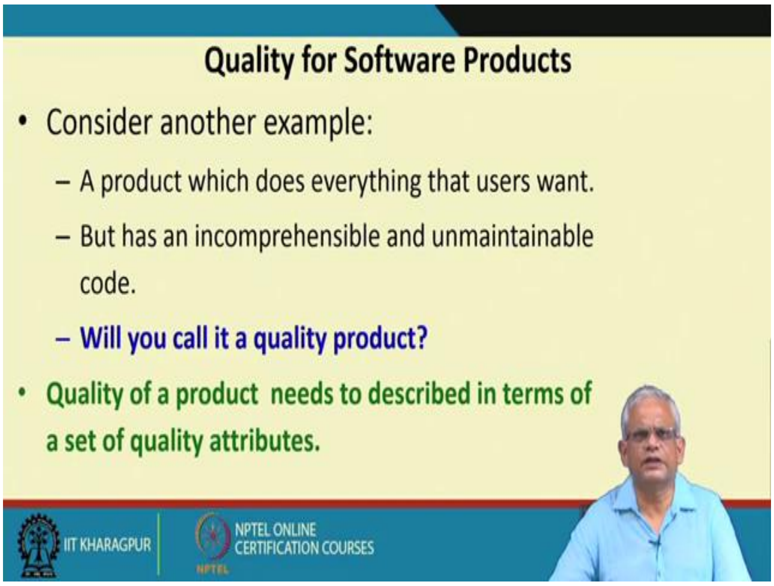
But to think of it satisfaction of the requirements is basically a users view; this is something similar to finding the quality of a car, table fan, food mixer, microwave oven etcetera. But then for software products; this is not really a good definition of quality, but then what is the difference between a table fan, food mixer, microwave oven and a software; why is it that fitness of purpose even though it is a good definition of a traditional product it is not a satisfactory definition of quality for software products?

(Refer Slide Time: 17:55)



The answer is that for software product, functional correctness is only one aspect of the quality there are many other aspects. For example, what if, its functionally correct, but users find it almost impossible to use very difficult user interface; we can't call it as a quality software.

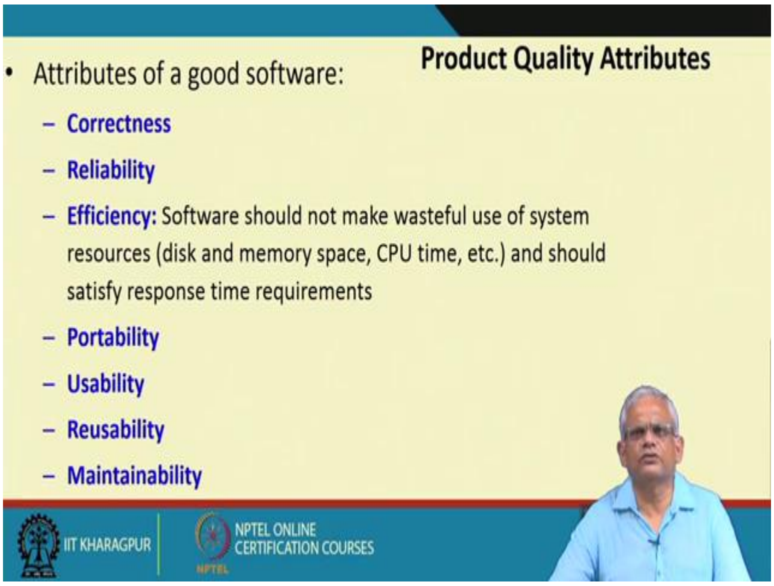
(Refer Slide Time: 18:31)



Similarly, another software it satisfies all the requirements that are written down in the requirement specification book, but then it has a incomprehensible and unmaintainable code. So, any bugs, any enhancements will be very hard to do for this software; let us say we installed a student grading software and find that after few days that there is a bug. Now, we want to maintain this software we requested that this bug be removed, but since it has a incompressible and unmaintainable code it will be hard to make any changes.

Similarly, if any enhancements are needed let us say all the functionalities were satisfied have been implemented correctly, but then we need a small enhancement, but it will be hard to do because the code is very bad we cannot call this as a quality software. So, one thing is clear that unlike other products; software products cannot be defined to be quality product based on just fitness of purpose or satisfaction of the requirements. We need to define the quality of a software in terms of a set of quality attributes. Of course, one of the will be correctness, but then there will be other attributes that need to be satisfied.

(Refer Slide Time: 20:41)



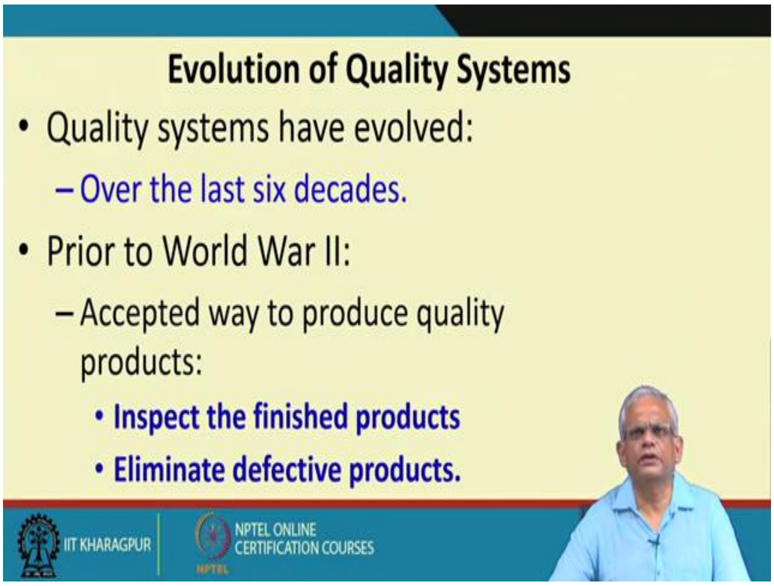
Let us look at what can be the set of quality attributes. Correctness is definitely one of the attributes for a quality software, reliability, chances of failure after uses. Efficiency it should not make wasteful use of resources for example, disk, memory space, CPU time and should satisfy the response time requirements.

Portability; tomorrow we want to run this on new hardware we should be able to easily do it that is also a quality attribute. Usability should have a good user interface the users should feel comfortable, easy to use. Reusability; if we want to extend this software, modify this for different purpose; it should be possible to do.

It should be possible to maintain the code should not be so bad that we cannot even maintain it. It should be possible to extend it, these are a possible set of quality attributes correctness, reliability, efficiency, portability, usability, reusability, maintainability, but there are can be many other quality attributes that may be considered.

Several practitioners and researchers have come up with quality models; the quality models basically try to estimate the quality of the software product based on some quality attributes; we will not really look at the quality models but we just need to be aware that based on these quality attributes; the quality models have been proposed so that given a software product we can measure the quality of the software product based on the quality model.

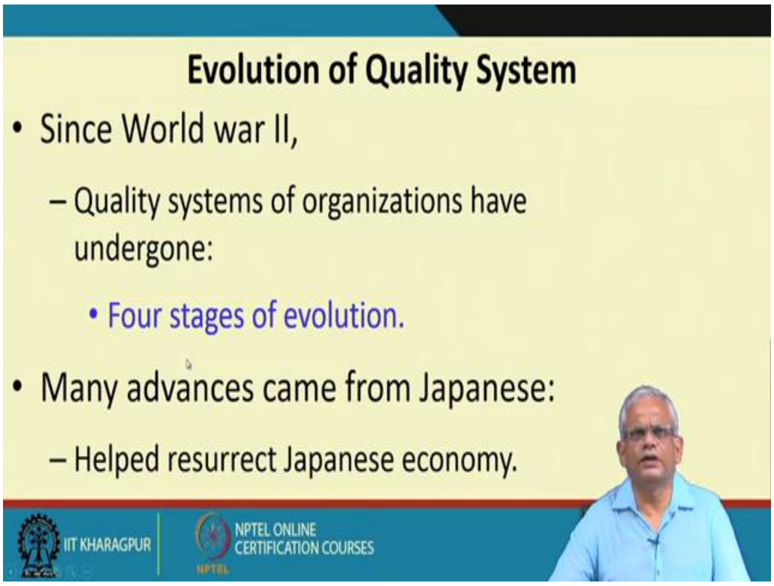
(Refer Slide Time: 23:11)



Now, let us see how the quality systems have evolved over years. In the last 6 to 7 decades; the quality system have evolved very rapidly; prior to the World War II; that is 1940s, 30s; it was accepted that to have a quality product, we must have a rigorous inspection of the finished products and then eliminate the defective products.

For example, a company manufacturing nuts and bolts; they would define a tolerance for the bolts and the nuts. And if they are beyond the tolerance just separate them out and reject them; the good product should be given to the customer and the bad one's are rejected. This was the only way that quality product where produced by various vendors before World War II; that is to rigorously inspect and test the product and then eliminate the defective product and give only the good products to the customers.

(Refer Slide Time: 24:53)



But then after the World War II, there have been four stages of evolution and may have come from the Japanese which helped resurrect Japanese economy and Japan is known to be producing very quality goods.

The initial quality efforts were testing that is identifying the defective products through testing; once that which fail the test are rejected good products are passed to the customer, bad products are rejected. But after that slowly we will see that the focus shifted from testing and rejection of the bad product to the process of manufacturing. We are almost at the end of this lecture, we will stop here and we will continue from this point in the next lecture.

Thank you.