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

Lecture – 07
Life Cycle Models – III

Welcome to this lecture, last time we were discussing about Life Cycle Models and we had said that the waterfall model is a intuitive and important model and many models have been derived from the water fall model.

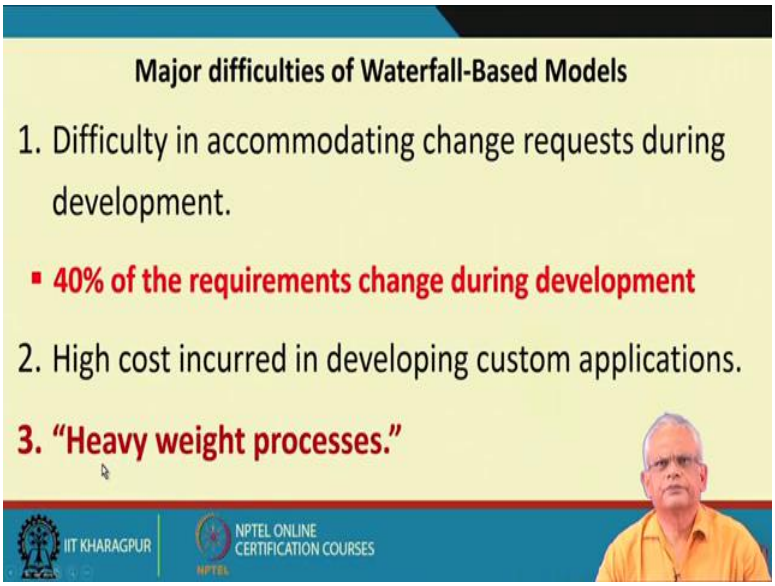
We had looked at the some of the derivatives from the waterfall model and today's lecture we will first see how the waterfall models have done, what are the good things about them and what are the short comings about them. And, then we will discuss about other models which overcome specific short comings of the waterfall model. There are some severe short comings of the waterfall model because of which they are not as popular as they were about 30-40 years back.

And one of the main problem that is being faced is that the type of project that are being handled are different from those that were handled 30-40 years back. Now, the projects are short and the code is not written from scratch because lot of code is available lot of code is being reused and only small modifications and tailoring is done which we called as service oriented software.

(Refer Slide Time: 01:58)

So, today we will first look at the some reflections on the waterfall based models and advantages disadvantages etcetera, waterfall model and we will see the models that have been proposed to overcome the specific short comings of the waterfall based models. We look at the incremental, the RAD and evolutionary model.

(Refer Slide Time: 02:23)



First let us look at the major difficulties of the waterfall based models. Possibly the most important difficulty is there is no way change requests can be handled in the waterfall model. In all waterfall based models that is the iterative waterfall model, the classical waterfall model, the prototyping model, the V model and so on. The requirements are gathered up front and these are documented, it is assumed that this will not change and then plan is made based on this documented requirements, design is done on this requirements and then coding and testing is done with respective these requirements.

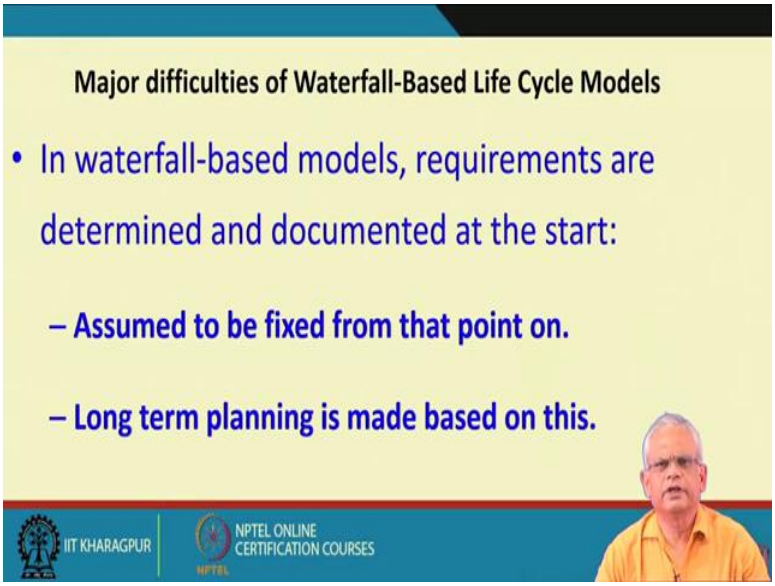
But then in reality, the present projects the requirements keep on changing as development proceeds. Typically about 40 to 50 percent of the requirements change after the initial requirement specification. And, this is possibly one of the major short coming of the waterfall based model, 40 percent of the requirement change during a typical development. The second problem is that many of the projects now are for customizing applications as we are discussing about the service oriented software where the organisation has some software and would tailor it for specific needs of another customer. If we use the waterfall model for this custom applications there will be huge

cost because the entire thing has to be documented, specification laid out, design and so on.

The third problem is that the waterfall model is called as heavy weight process, it is called a heavy weight process because lot of documentation is produced as part of the process, everything is supposed to be documented. There is starting from requirement specification, the reviews, review results, the design various types of plans everything is to be documented and that is the reason these are called as heavy weight processes. And it is estimated the typically 50 percent of the effort by a development team goes towards creating documents and naturally the project costs increase and there is project delays.

And specifically now that the project durations are very short such heavy weight processes are not finding favor and we need some processes which do away with these problems.

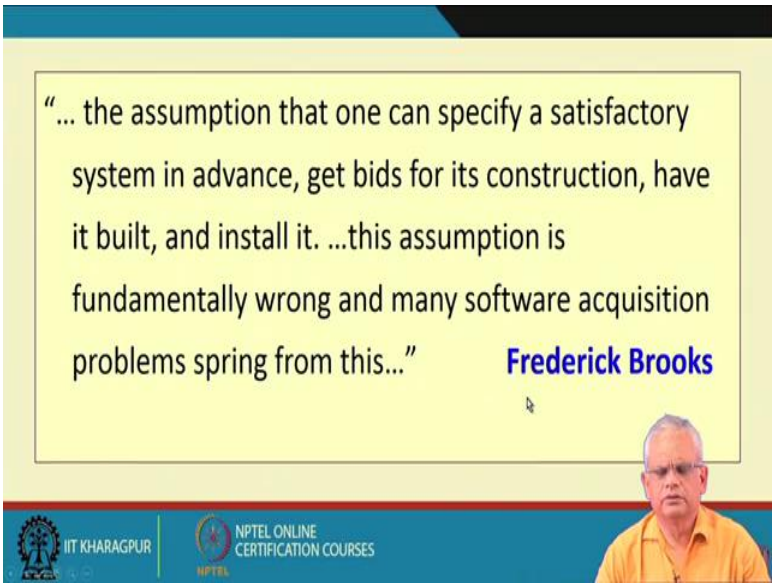
(Refer Slide Time: 06:03)



Let us just do a bit of elaboration that in waterfall model, requirements are determined and documented at the start, they are fixed from that point on; no changes are permitted and long term plans are made based on these documented requirement. The design is made on based on these documented requirement even test cases are written based on this documented requirement.

And therefore, any change to this requirement will require changes to all documents and therefore, those who use the waterfall model they heavily discourage the customer to make even the small changes.

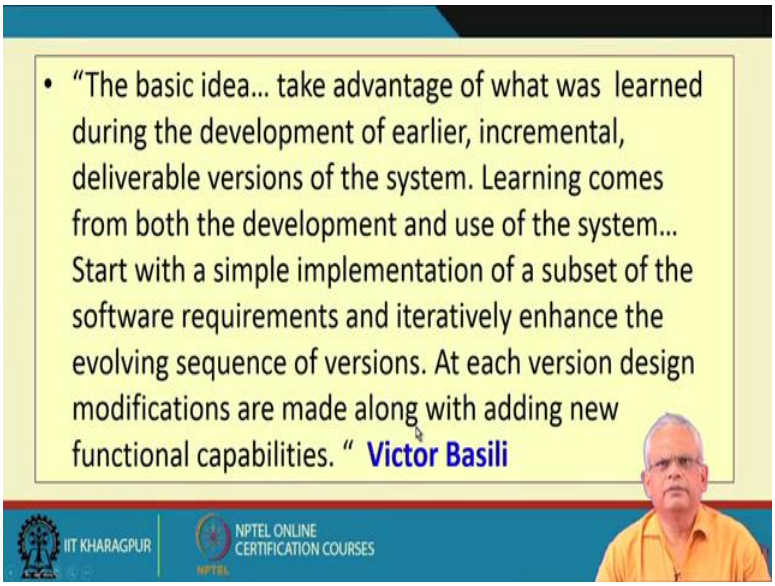
(Refer Slide Time: 07:00)



But let us look at the observation of Frederick Brooks who is one of the guru's in this area. The assumption that one can specify a satisfactory system in advance, get bids for its construction, have it built and install it. This assumption is fundamentally wrong and many software acquisition problems spring from this.

Just to explain Frederick Brooks thought, he felt that many of the project failures, project delays etcetera are the result of having the requirements worked out at the beginning of the development. And, then all the development proceeds based on the documented requirements is fundamentally wrong and many of the problems arise from this. In response to this short coming of the waterfall model one model that was proposed is the incremental model. To get the main idea behind the incremental model, let see what Victor Basili one of the founders of this area has to say.

(Refer Slide Time: 08:24)

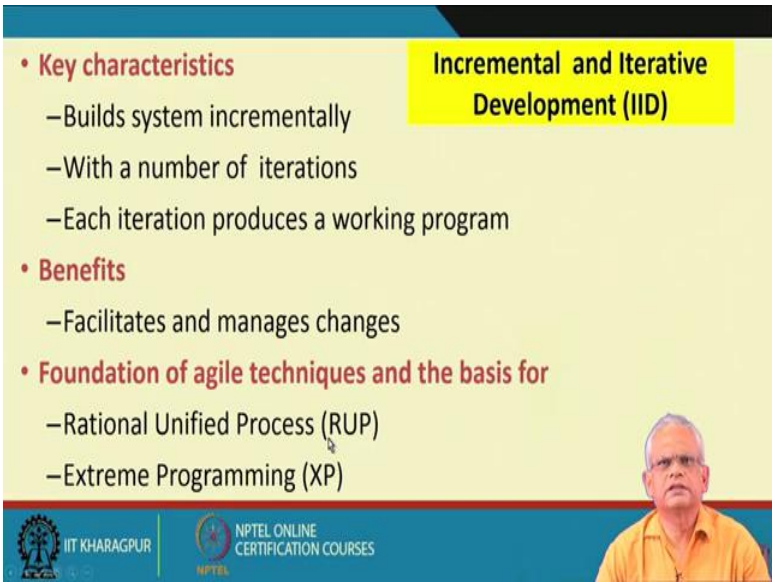


The basic idea is to take advantage of what was learnt during the development of earlier, incremental, deliverable versions of the system. Learning comes from both the development and use of the system. Start with the simple implementation of a subset of the software requirements and iteratively enhance the evolving sequence of versions.

At each version design modifications are made along with adding new functional capabilities. Thus to explain what Victor Basili has to say about a incremental model is that, the software is developed in increments. In the first increment some of the core functionalities are implemented and then these are given to the customer who gives feedback and this forms a learning from the use of the system, even the developers they have they learn from the implementation of the core functionalities. And based on the feedback the system the software is refined and this is called as the iteration.

In iteration the same functionality is refined each time the customer gives new feedback, the same functionality gets refined iteratively and also at the same time new functionalities are also implemented these are called as increments. So, this model is called as incremental model with iteration.

(Refer Slide Time: 10:47)

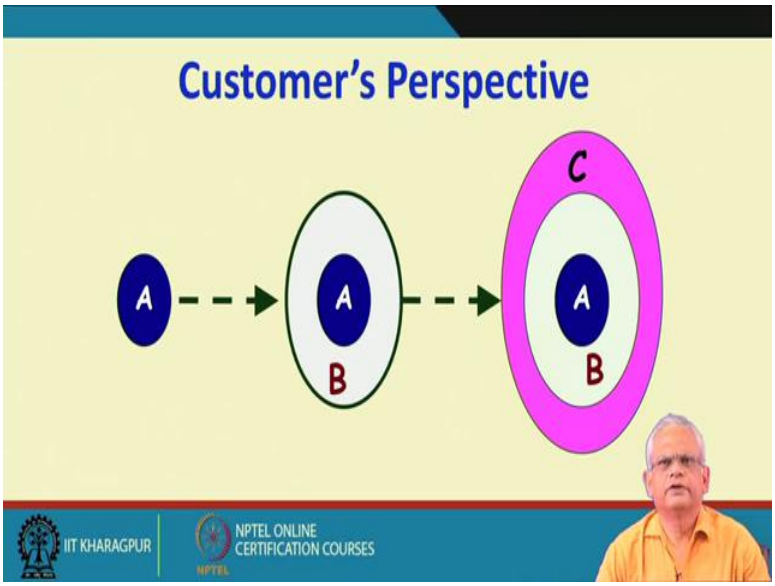


Incremental and iterative development here each time new increments of functionalities are implemented and at the same time the functionalities that were already implemented they are refined and that is called as iteration with those functionalities.

The key characteristics of the incremental and iterative development is that, built the system incrementally small parts of the system are built. There are number of iterations for each functionality and each time a deliverable is given to the customer in the form of a working program which the customer can use and give feedback.

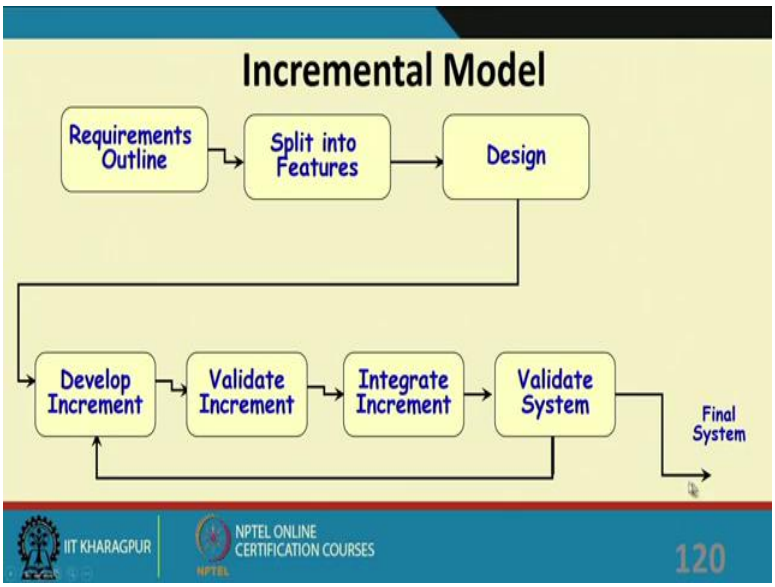
Each deliverable to the customer would not only have incorporated the feedback that he had given in the previous increment, but also added new functionalities. As can be easily seen this is a good way to manage changes, the customer is encourage to feedbacks and these are incorporated. The incremental model has become popular in industry, the rational unified process is a incremental model, the extreme programming the agile models etcetera are all incremental.

(Refer Slide Time: 12:29)



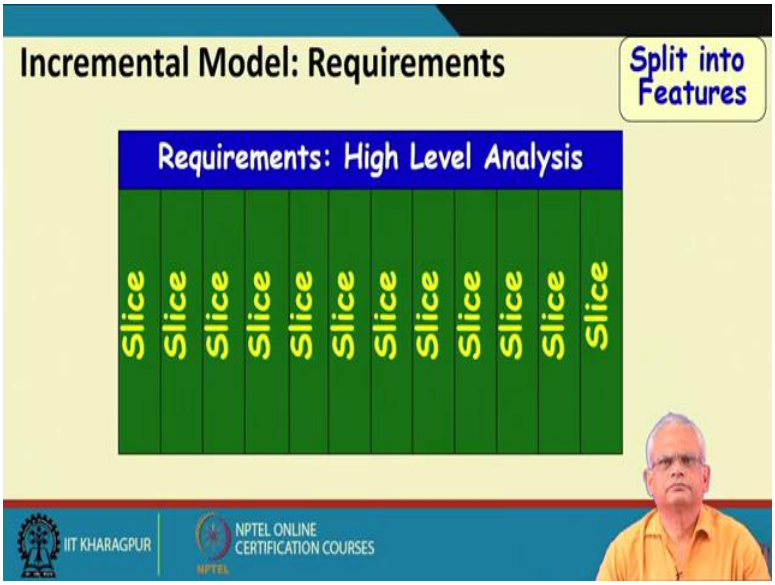
From the customer prospective gets a functionalities some part that he can use and give his feedback based on this the organisation developing organisation incorporates the feedback on the core functionality and also implements additional functionality and gives it to the customer. The customer again gives feedback on this and the feedback is incorporated and also new functionality are added and this way the system gets completed and hopefully at the end the delivered system will meet the customer's requirements because all his feedback in using the system have been incorporated.

(Refer Slide Time: 13:27)



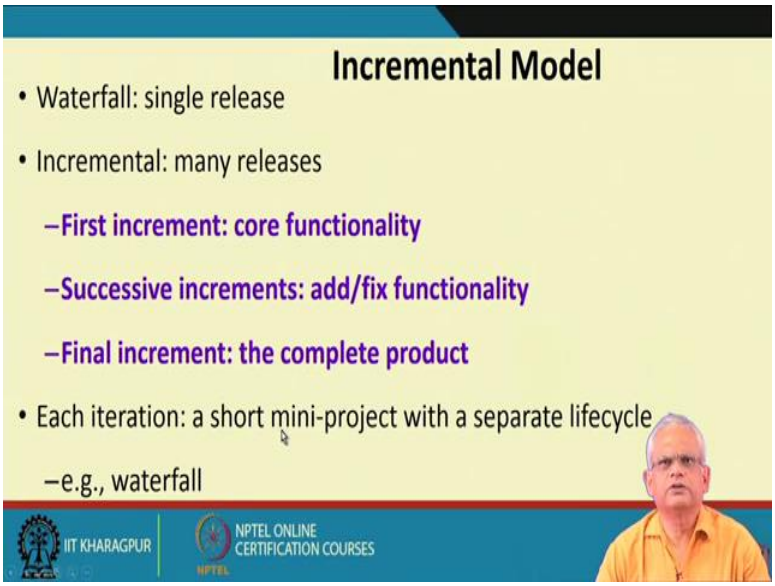
If you can diagrammatically depict the model initially requirements are collected and this requirements are split into incremental features there is a overall design and then each time one increment is taken developed, validated, integrated with previous increment, the system is validated and delivered to the customer feedback is taken next increment is developed and so on until the final system is updated.

(Refer Slide Time: 14:16)



As you can see that in the increment model the requirements are collected up front these are split into small increments or slices. Similarity, here with the waterfall model is that the requirements are collected up front, but the difference is that each of this increment customer feedback is taken and these change.

(Refer Slide Time: 14:58)

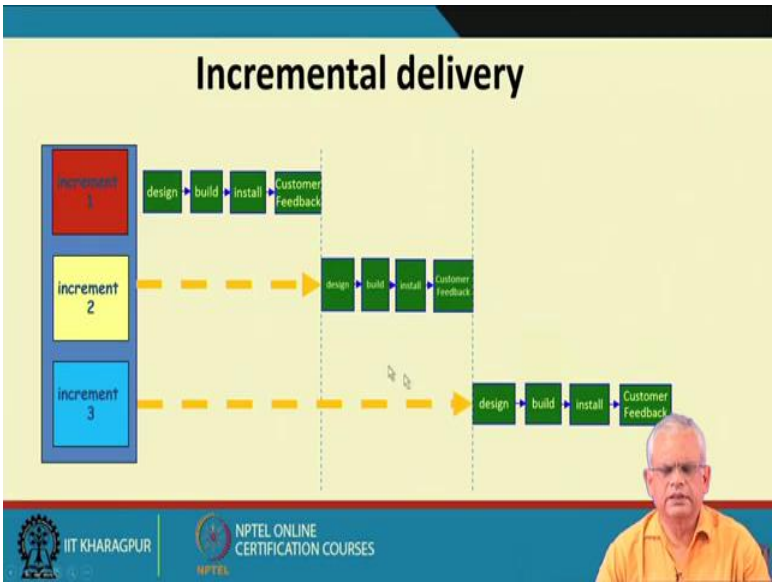


In a waterfall model there is a single release initially all the functionality requirements are collected, designed, coded, tested and given to the customer as a single release whereas, in incremental many releases are there and in after each release customer feedback is taken and the functionality that was already delivered are again changed to meet the requirements of the customer.

The first increment is typically the core functionality, then the successive increments they not only refined the existing functionality, but add or fix functionality the final increment is the complete product. This is called as the incremental model with iteration, but then somebody can think of a purely incremental model where there are no iteration in which the system is just developed in small increments and given to the customer, but the customer feedback is not really taken to change the existing functionality there is no iteration.

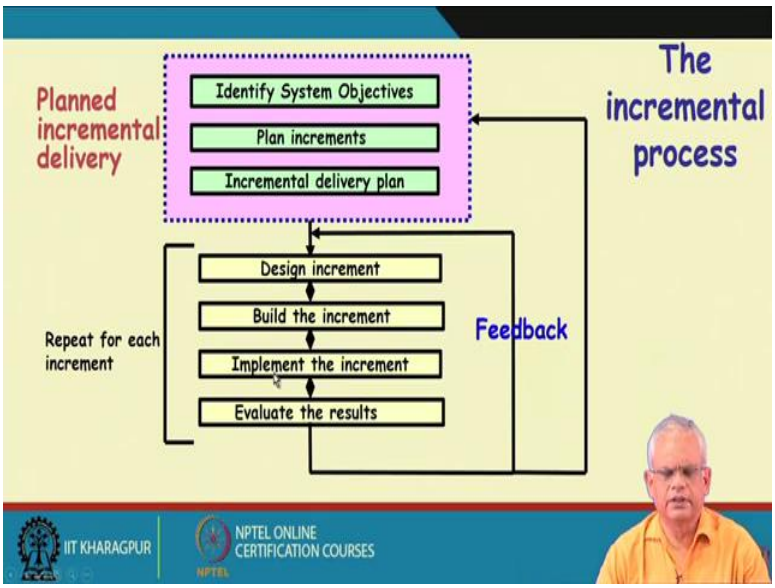
In a purely incremental model the software is developed in increments and delivered to the customer, but once the functionalities are delivered these are not changed, but in a incremental model with iteration, the functionalities are delivered to the customer in increments. But, also the customer feedback is taken after each functional functionality is delivered and those are incorporated in the next iteration. Of course each increment or iteration is a small is a sort mini project with a separate life cycle for example, for each small increment may be a waterfall model can be used.

(Refer Slide Time: 17:24)



This is a diagrammatic representation, the initial requirements is split into increments increment 1, 2, 3; increment 1 is developed, increment 2 gets developed. So, this is like a waterfall model design built install customer feedback. Increment 1 is first developed, then increment 2 is developed, then increment 3 gets developed.

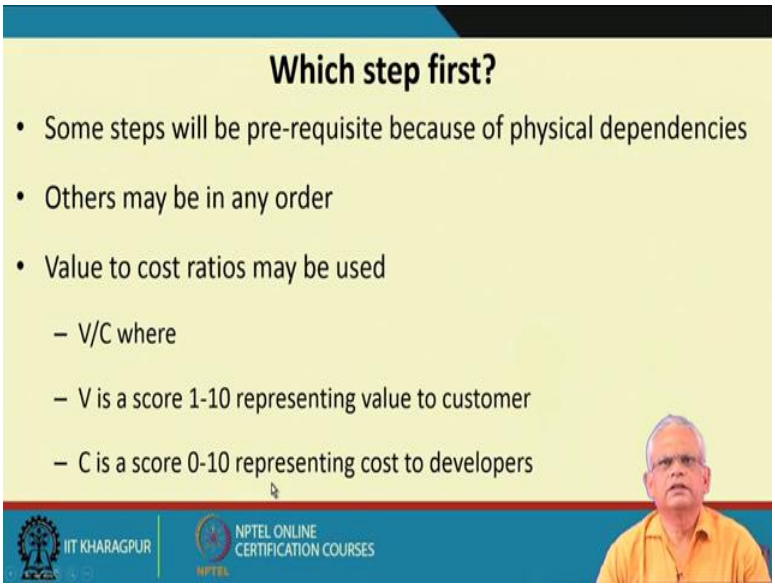
(Refer Slide Time: 18:04)



If we diagrammatically represent this model of development first we gather all customer requirements document them and then plan the increments and also the incremental delivery plan.

Then design the increment, build the increment, implement the increment and then give the customer for evaluation the increment and then use that as a feedback and take the next increment and so on.

(Refer Slide Time: 18:47)

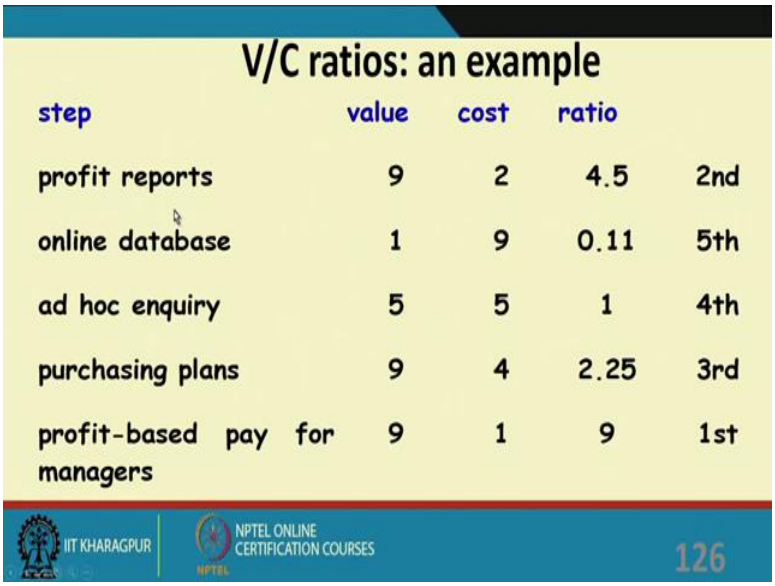


Because there are several increments out of which the designer need to choose one of the increment for delivery in the next iteration, which increment should be selected first? This is an important problem.

Some parts will be pre requisite of the other parts without that the system will not work. For example, until the customer registration is done customer billing cannot be done first we must have the customer registration software developed and then only the customer billing software can be implemented, but then many of the increments are without any order.

Now, these increments where there are no order value to cost ratio may be used, this we call as the V by C ratio, where V is the value to the customer which is graded in a scale of 1 to 10 and C is a score 0 to 10 representing the cost to the developers.

(Refer Slide Time: 20:24)

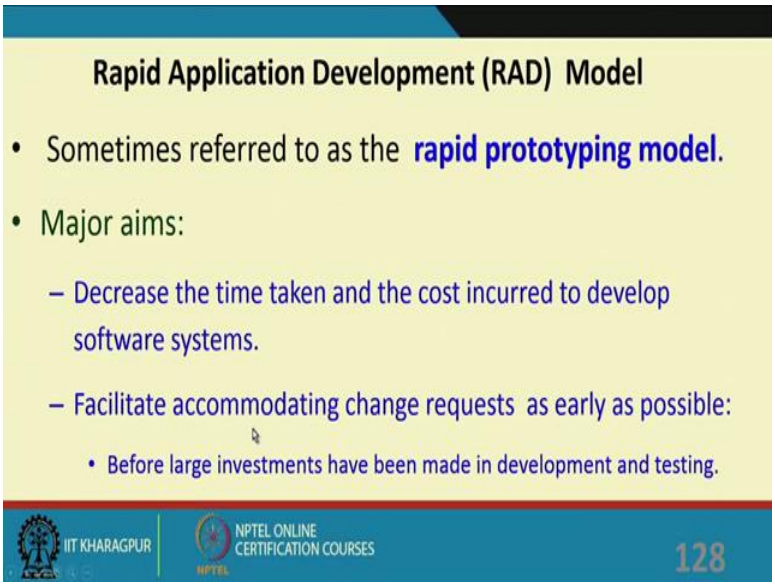


Let us take an example to see how it works. Let say at one step in increment for the next increment we have let say these are the modules that the development team has the flexibility to implement profit generate the profit reports, implement the online database, implement ad hoc enquiry, implement the purchasing plans, implement the profit based pay for the managers. These are the five functionality, that the developers have the choice and from the customer feedback about the value of these different functionalities they have assigned these values.

Online database is the lowest value to them profit report is 9, ad hoc enquiry 5, purchasing plan 5 and profit based pay for manager is 9. The development team when ask to rate this on 1 to 10 I found that the least cost for development is profit based pay for manager, profit reports is 2, purchasing plan is 4, ad hoc enquiry is 5 and the most difficult to implement at the online database and then on computing the V by C ratio we find that there is a scale of 9 to 1, 0.1; 0.11.

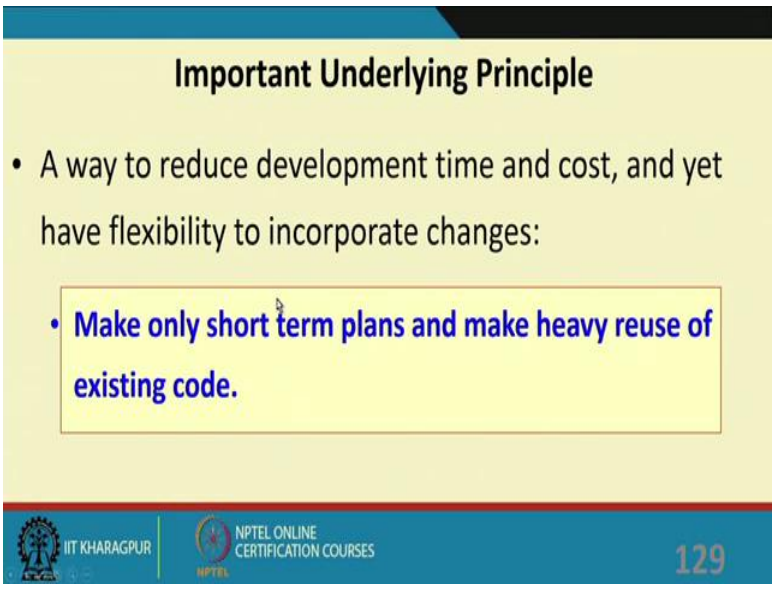
So, for the next increment this is one that should be chosen this as high V by C ratio and n the profit report should be chosen, third is the purchasing plans and forth is the ad hoc enquiry and last should be the online database. The incremental model as become extremely popular, it is almost part of every model that is being used those are derivatives of the incremental model one important derivative is the RAD model, RAD stands for Rapid Application Development.

(Refer Slide Time: 23:04)



This is sometimes called as the rapid prototyping model; here the main aim of the model is to decrease the time taken and the cost incurred to develop the software. Facilitate accommodating change requests as early as possible before any large investments have been made.

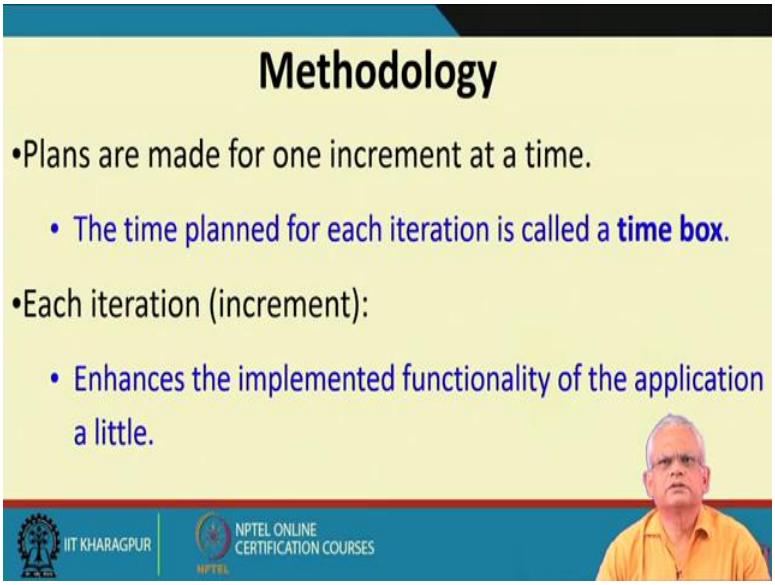
(Refer Slide Time: 23:29)



This is as we said is a incremental model here increments are decided and only short term plans are made and heavy use of the existing code. Contrast to the waterfall model this model is very efficient, it can incorporate flexibility to incorporate changes because

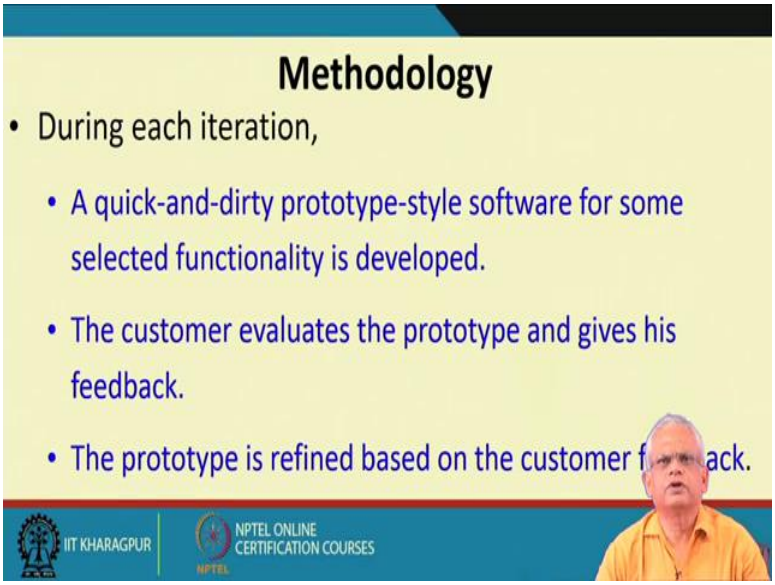
only short term plans are made any change does not require changing all the plans that is design the project management plan etcetera, the test plan and so on.

(Refer Slide Time: 24:22)



The plans are made for one increment at a time and this increment is called as a time box and after each increment is delivered, the system or the applications becomes more complete.

(Refer Slide Time: 24:47)



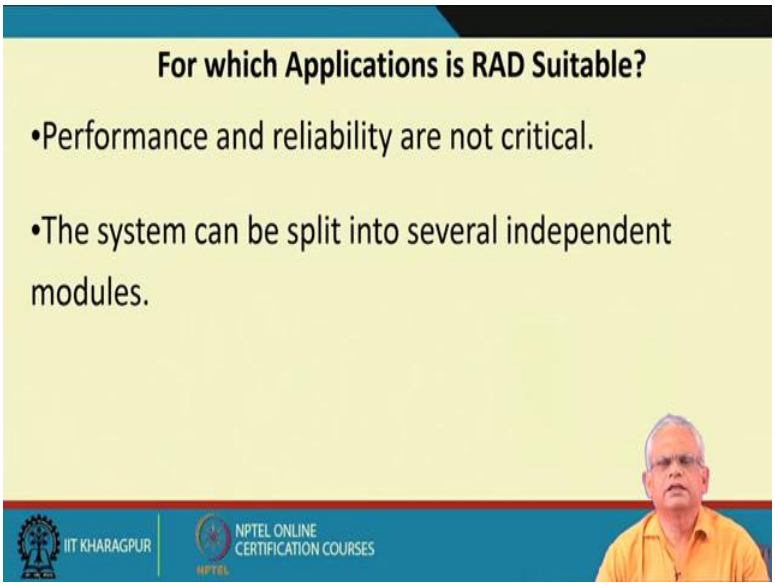
During each iteration a prototype of the software is first developed is given to the customer for evolution and then this prototype is refined based on the customer

feedback, one point to note here is that in this model the prototype itself is refined in the RAD model the prototype itself is refined to the actual to be the actual software, where is in the prototyping model that we had discussed which is a derivative of the waterfall model. The prototype is used to get customer feedback the start of the project and then the prototype is thrown away and the software is developed fresh. But in the RAD model the prototype is refined based on the customer feedback.

(Refer Slide Time: 25:39)

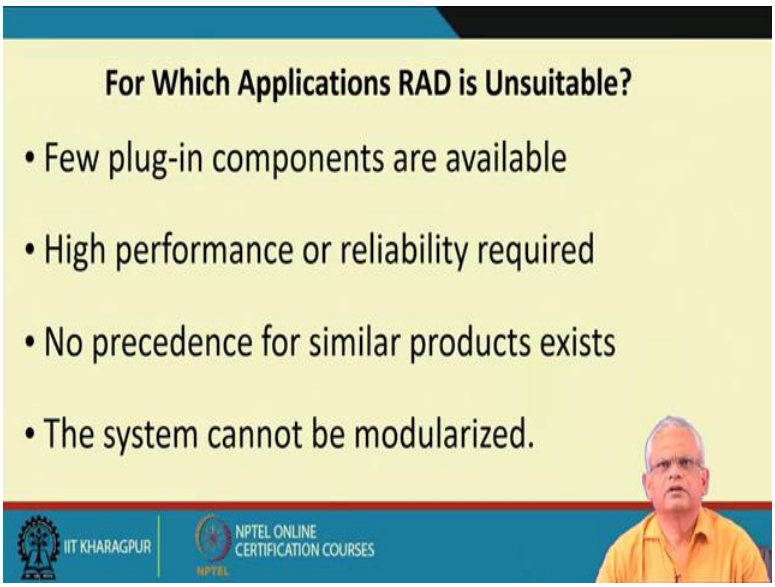
The RAD model achieves faster development, it encourages use of specialized tools, the tools should be supporting visual style of development like drag and drop. Use of reusable components and use of standard APIs these are something that are suppose to help the software implemented faster.

(Refer Slide Time: 26:20)



Now, let us look at the software projects for which the RAD model is suitable. The projects for which the performance and reliability are not critical, but these are required to be developed very fast and the system can be split into several independent models. If the system is very trivial and we cannot have several increments then; obviously, RAD model is unsuitable.

(Refer Slide Time: 26:49)

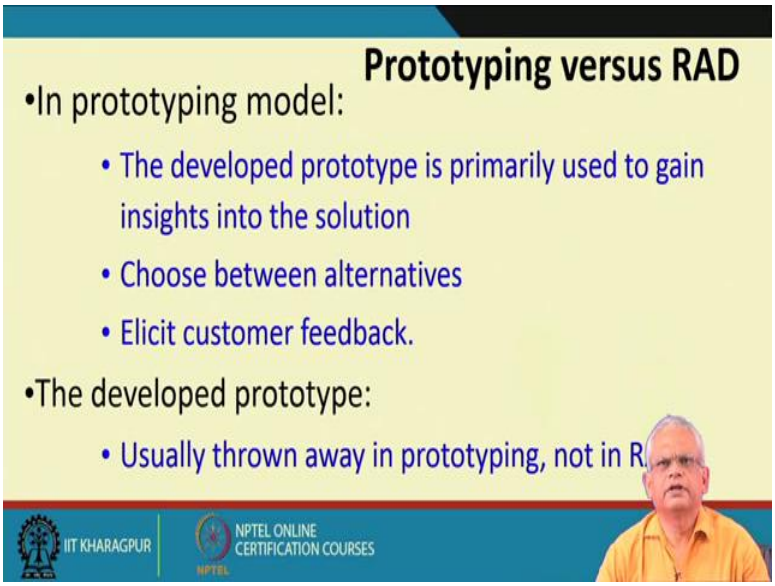


Also the RAD model is unsuitable when we have few login components are available, high performance or reliability is required. The reason why the RAD model does not

give high performance or reliability is that the prototype itself is refined into the actual software. Unlike a waterfall model where a fresh software is designed coded and tested, the RAD model would give a low reliability and performance.

If there are no precedence for similar products then also RAD model is unsuitable that is challenging new type of software for which; obviously, there are no components would be available to use as plugins the systems that cannot be modularized RAD is unsuitable.

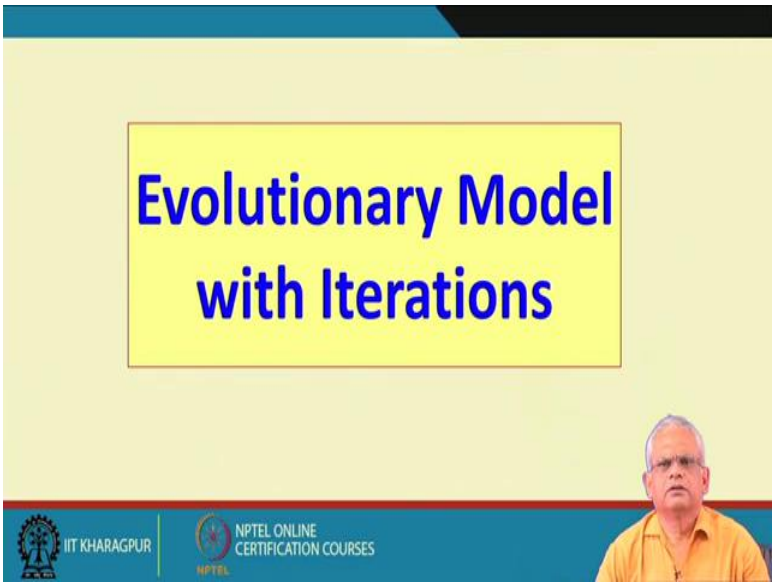
(Refer Slide Time: 28:01)



The RAD model each time constructs a prototype and gives to the customer for evaluation and based on the customer evaluation the prototype is changed and as the customer gets satisfied the refined prototype is becomes the part of the software. In the prototyping model which is a derivative of the waterfall model, the developed prototype is primarily used to gain customer feedback and also to get insight into the solution that is the technical issues and so on.

And therefore, using the prototyping model you can chose between different design alternatives, you can elicit customer feedback and the developed prototype is usually thrown away, but not in the RAD model. In RAD model the prototype itself after enhancement refinement becomes the final software.

(Refer Slide Time: 29:18)



Another important development model is the evolutionary model with iteration, we had seen the incremental model with iteration which is of course, important and has become part of many development models that are being used across industry, but another idea that has come is the evolutionary model with iteration. We will examine this model in the next class and we will see that the present model that has become extremely popular at the agile models which have the features of both the incremental and evolutionary model with iterations.

We will stop here and continue in the next lecture.

Thank you.