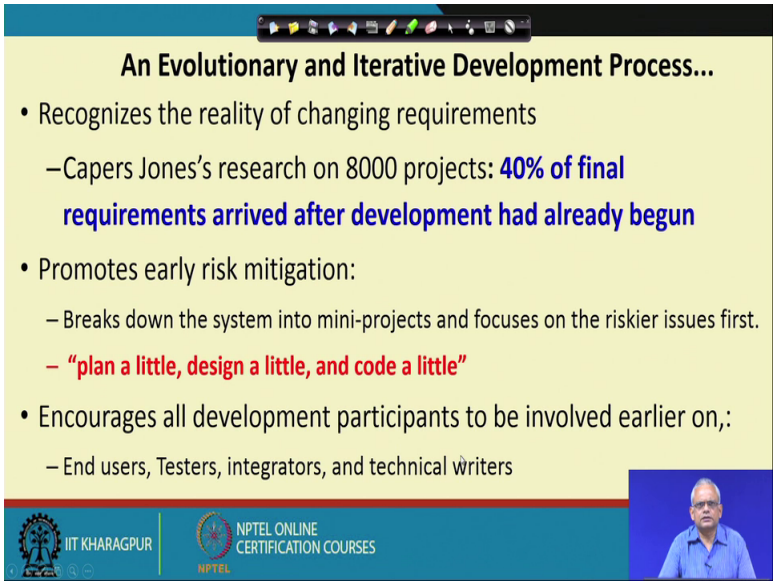
Software Engineering
Prof. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 11
Evolutionary Model

Welcome to this lecture. In the last lecture, we had discussed about the Incremental Development Model. The Incremental model overcomes many shortcomings of the waterfall model. This lecture, we will look at the Evolutionary model with iterations.

(Refer Slide Time: 00:43)



In any real project, the requirements change during the development. There are many reasons why the requirements change? One is that the customer is not able to view the software in entirety before the software is built. And therefore, many requirements are missed; many requirements are given ambiguously; there are inconsistency. Capers Jones's an established researcher. He studied 8000 projects and found that 40 percent of the requirement change during the development.

This is a very significant issue because neither waterfall model nor the incremental model handled this satisfactory. Waterfall model has no provision for requirement change during development. Incremental model, if we recollect the previous lecture involves identifying all the requirements upfront and then slicing the requirements into incremental features to be deployed at the customer side. Of course, while a feature is

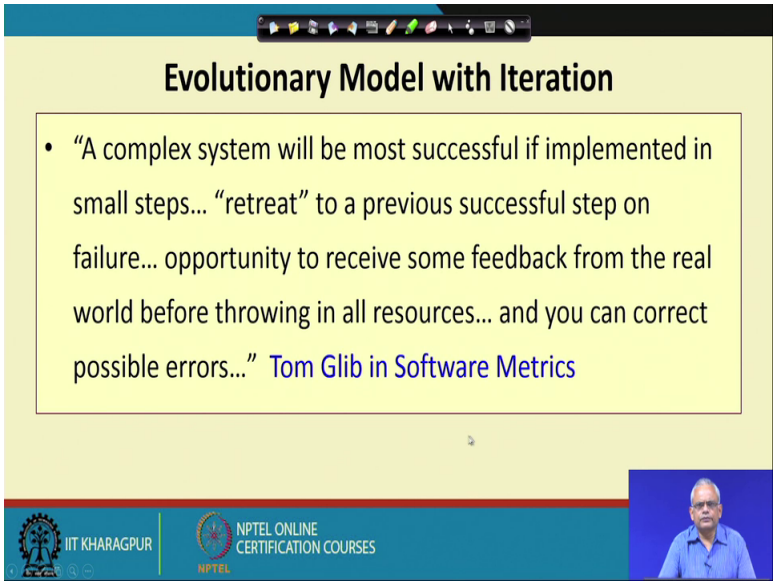
being installed the customers feedback is obtained which may necessitate additional requirements or change of requirements.

But, still the fact remains that incremental model requires all the requirements to be identified upfront as far as possible. Let us look at the Evolutionary model which tries to overcome this necessity of having to identify all the requirements upfront. There are some similarities between the evolutionary model and the incremental model in the sense that increments are deployed at the client side.

But here, in the evolutionary model no upfront requirement specification is required. Initially some features are implemented which have been identified. And then, as the customer experiments which those more and more features get developed and deployed. So, the software actually evolves starting with something very simple, but the main difference is the incremental model, and the purely evolutionary model is that with respect to upfront identification of the requirements.

The evolutionary model also is called as a "plan a little, design a little, code a little" because each time only few features are identified. Plan for those feature development is only done those are designed and coded and deployed at the customer side. This model just like the incremental model involves the end users, they get frequent religious; they can give their experience on that. The testers, incrementers, technical writers all get involved from the start of the project unlike the iterative waterfall model.

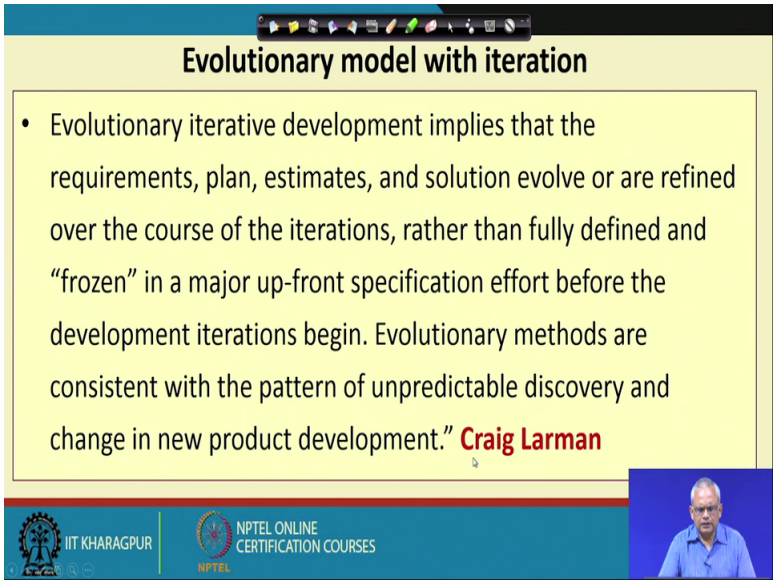
(Refer Slide Time: 04:48)



Let us see what Tom Glib an eminent personality in this area has to say. “A complex system will be most successful if implemented in small steps, “retreat” to a previous successful step on failure, opportunity to receive some feedback from real world before throwing in all resources, and you can correct possible errors”

So, he has identified the big advantage of the revolutionary model that the client feedback is obtained and the features are determined as the client keeps on using the system evolves. And if some feature the customer does not like, then that feature is simply discarded and new features replacing that are built.

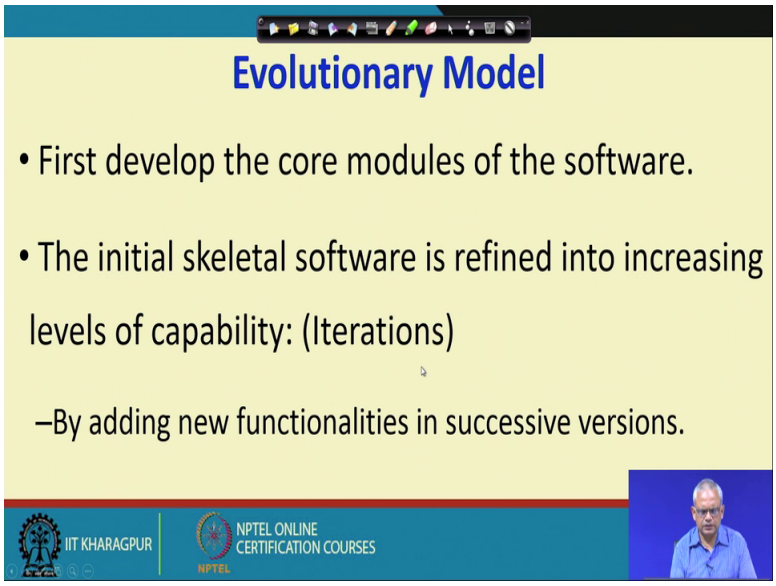
(Refer Slide Time: 05:53)



Craig Larman another celebrated personalities in this area, let us see what he says? Evolutionary iterative development implies that the requirements, plan, estimate, and solution evolve or a refined over the course of the iterations, rather than fully defined and “frozen” in a major up-front specification effort before the development iterations begin. Evolutionary methods are consistent with the pattern of unpredictable discovery and the change in new product development.”

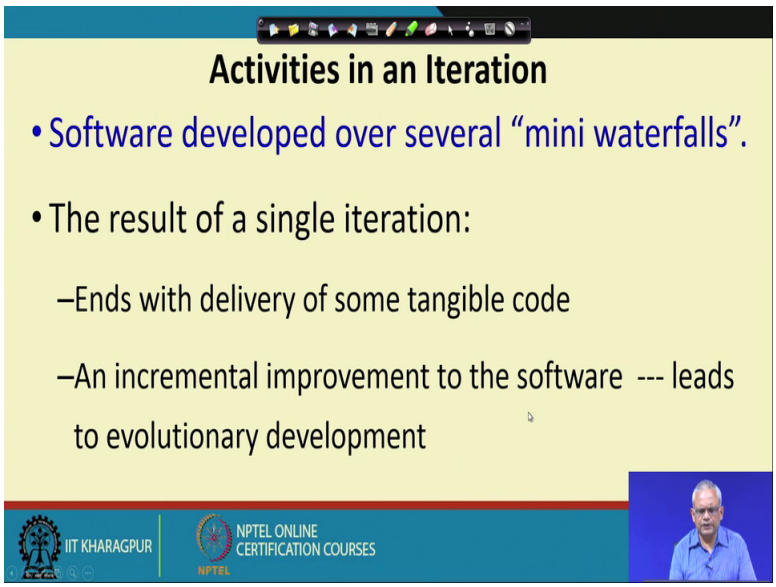
So, Craig Larman also agrees that upfront specification is very difficult and evolutionary model is bound to be much more useful in real life projects.

(Refer Slide Time: 06:54)



Let us look at what is involved in the Evolutionary model. Based on a overall understanding of the software, first the core modules of the software are developed and this core module are refined into increasing capability levels which are called as the iterations.

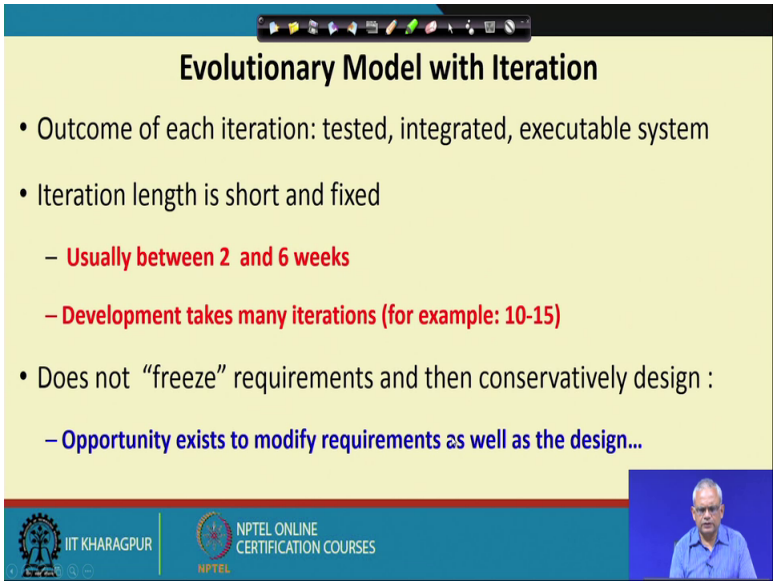
(Refer Slide Time: 07:28)



Each iteration adds new functionalities and these are successively deployed at the client site. Each iteration is actually a mini waterfall; after a single iteration some code is

deployed and the client expresses what is required next and this leads to the evolutionary development.

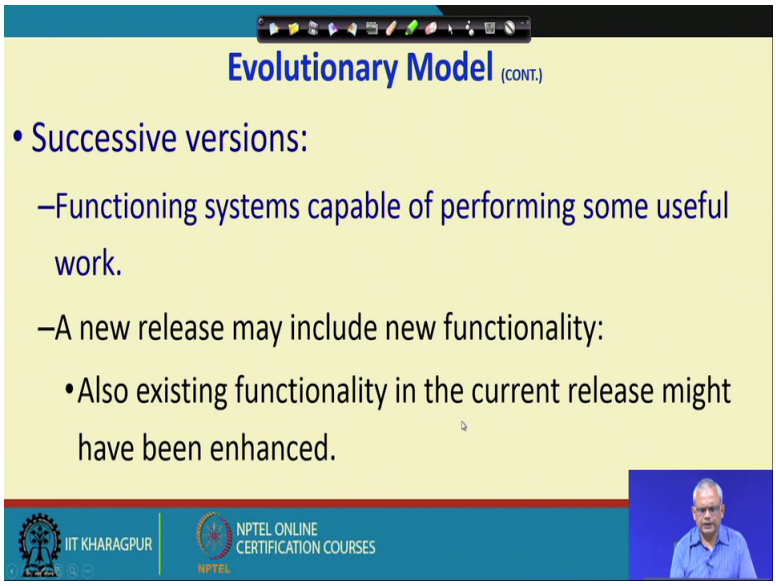
(Refer Slide Time: 07:56)



In each iteration some code which is tested, integrated, executables system is deployed at the client side. The iteration length is sought typically 2 to 6 weeks. A software to be completely developed may take somewhere between 19 to 15 iterations.

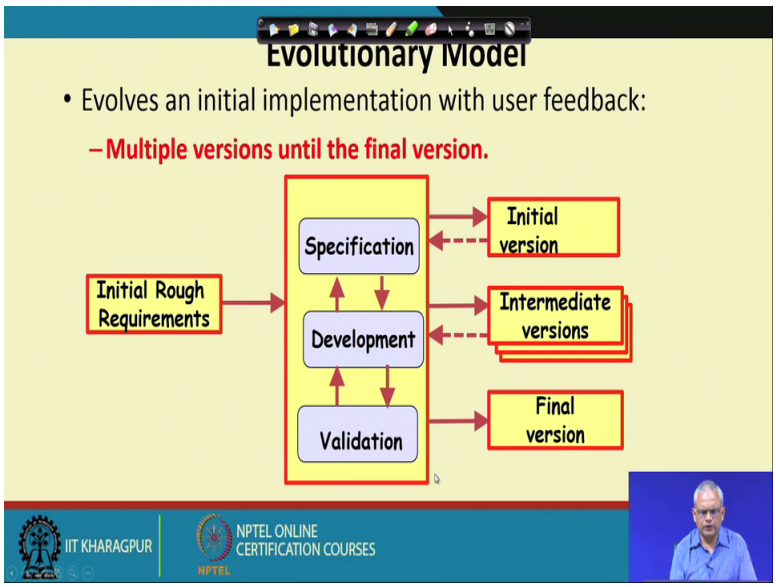
And here, as we have emphasized several times that requirements are not collected upfront and frozen and then, conservatively designed to accommodate any future changes. But here, the requirements are allowed to change and the design also changes.

(Refer Slide Time: 08:52)



The successive versions these are fully functional systems. The customer can make use of them in real work, when a new religious made new functionality are added. But some of the existing functionality based on the customer feedback might have been changed or enhanced.

(Refer Slide Time: 09:24)

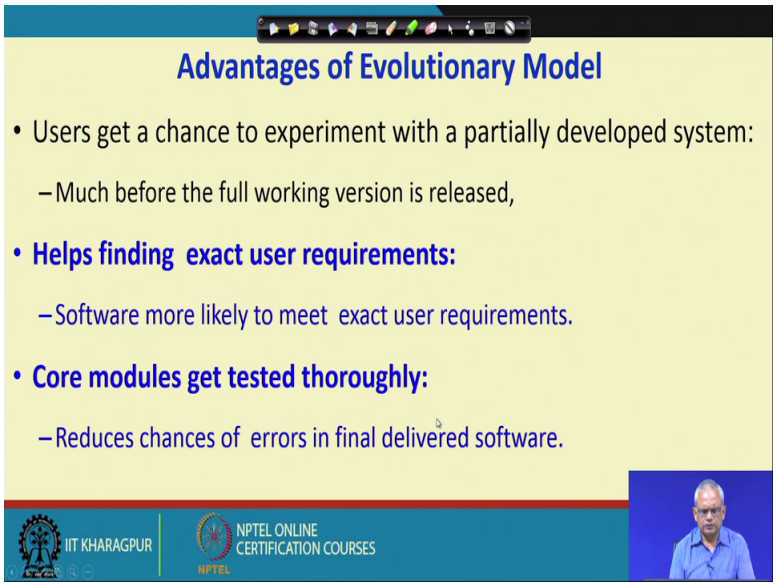


We can represent the model in this schematic; initially some rough understanding of the system is done, it is not full specification of the system unlike the incremental model.

Here just overall feature required and then the core and this is the iteration, every iteration there is a specification development validation and then finally, the deployment.

The core is deployed and the dotted arrow shows feedback obtained. And again, the iteration occurs successive versions are deployed at the customer site; feedback obtained and this goes on the iterations goes go on until the final version is deployed at the customer site.

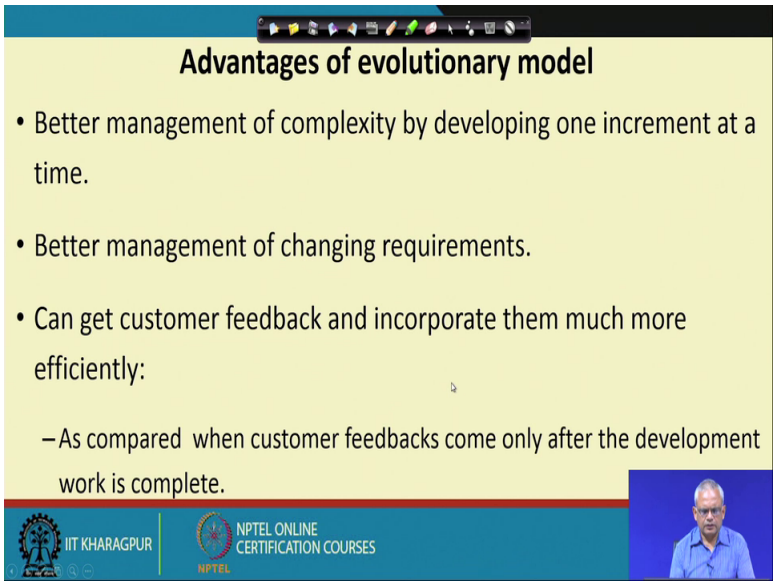
(Refer Slide Time: 10:30)



There are many advantages of this evolutionary model. The users can experiment, give their feedback; they use it for real work. So, that they get a real feel of how it performs. And therefore, it helps find exact user requirements, and finally, when it is deployed at the customer site very likely to meet the exact user requirements.

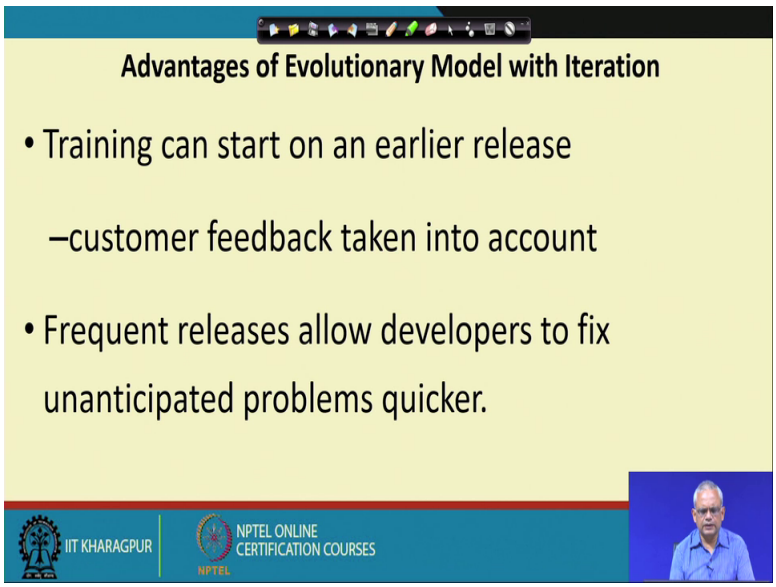
And also since the modules are deployed and used at the customer site any defects are noticed and reported. So, the core modules are used for long time and all defects must have been detected. And therefore, the final delivered software is much more reliable than would be the case in the waterfall model.

(Refer Slide Time: 11:39)



Since, at a time the developers focused only one or two features; the complexity is easily manageable. It welcomes changes from the customer after the use and is incorporated. No long term plans are made and that is the region where the changing requirements can be easily accommodated unlike the waterfall model.

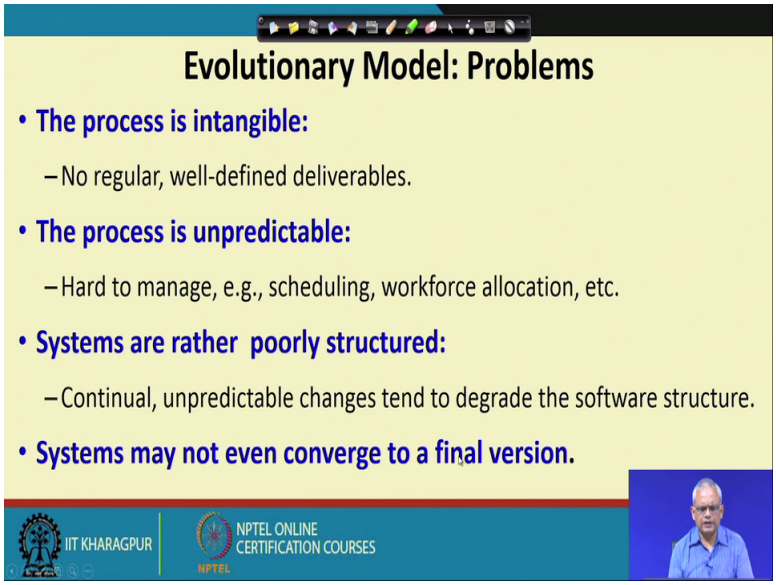
(Refer Slide Time: 12:16)



The customer deploys and can incremental learn the software and also any part that is difficult to use feedback is obtained and made much more usable. Frequent religious allow the developers to fix any problems quicker because they are into the development

and any problem that occurs they developed something very recently they can fix it quickly.

(Refer Slide Time: 12:56)

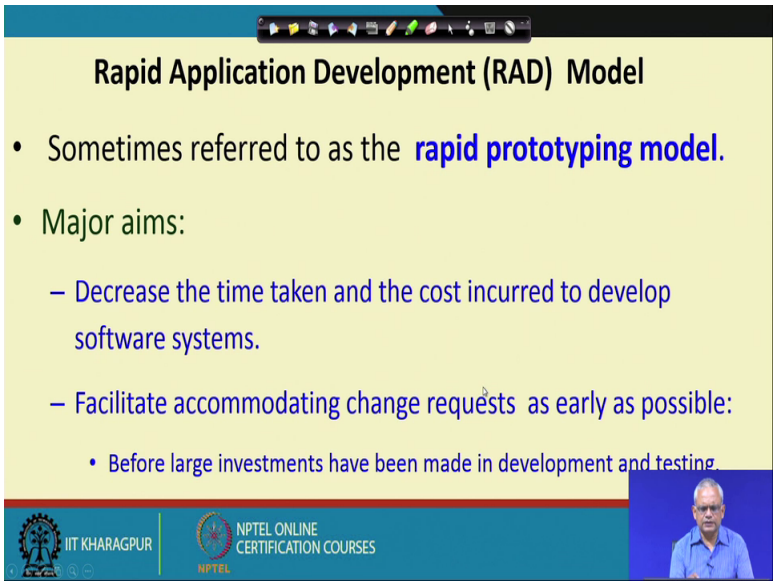


But then, the evolutionary model has it is problem. One is that the process as we described our edge is described in the literature is rather vague. And therefore, the process is intangible. No regular well define very deliverables only the incremental deployment that is required; the process is unpredictable.

Because no long term plans are made, we do not know how long a software will take once we start the iterations. We do not know how many iterations it will take. It may complete in 3 months; it may complete in 6 months. So, if the control is very less. From the beginning it is hard to predict how many iterations; it is hard to manage the workforce, how many persons will work, for how long cost and so on.

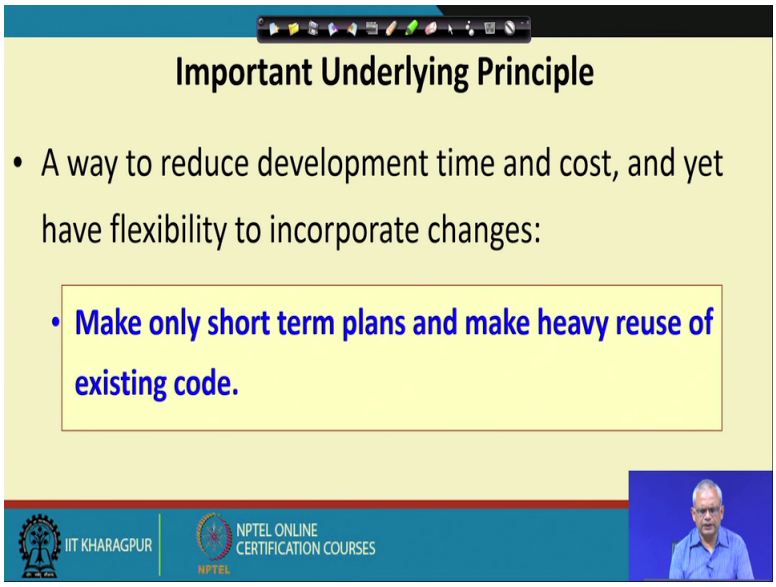
Another major difficulty here is that the software keeps on changing. And therefore, it typically has a poorly structured design Continuous changes to the software degrade the design of the software and also it may so happen that the clients keep on giving features modifying features and so on. And in the bad case the system may not even converge to a final solution, the project may not complete.

(Refer Slide Time: 14:57)



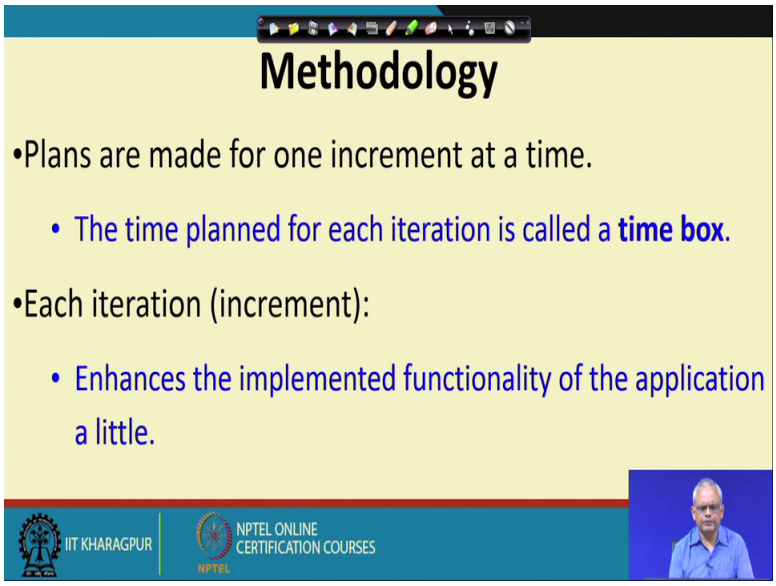
Now, let us look at the RAD model. The RAD model stands for rapid application development model. This is also a popular model; it is called as the Rapid prototyping model. As the name rapid here indicates that the emphasis here is to reduce the development time and the cost and also it facilitates accommodating changes as early as possible.

(Refer Slide Time: 15:28)



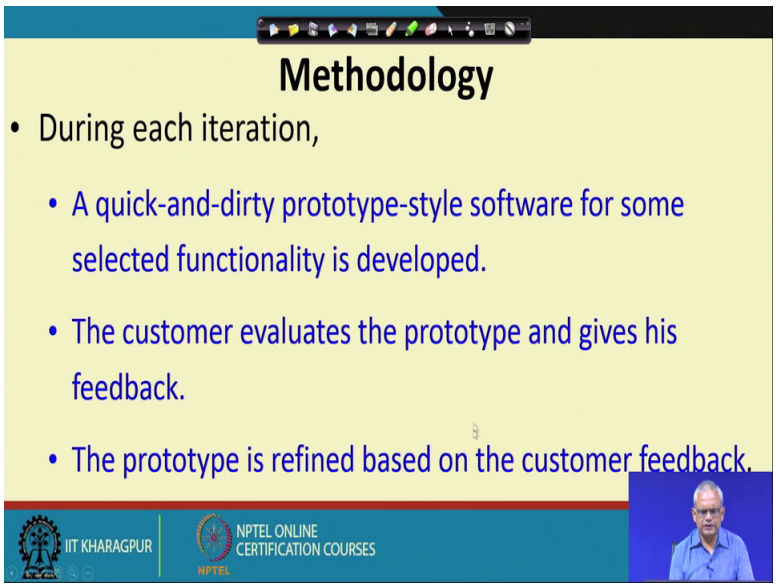
Here only short term plans are made and another feature is heavy reuse of existing code that is customization projects.

(Refer Slide Time: 15:47)



Here again it is a incremental development methodology that at anytime only one increment is planned, and over one iteration it is completed, and deployed, and the iteration is called as a time box here. Each iteration or increment enhances the functionality of the application little.

(Refer Slide Time: 16:17)



Unlike the incremental and the evolutionary model, in the rapid development model since the focus is to develop something very rapidly. Therefore, a quick and dirty prototype is developed and deployed at the customer site to obtain the feedback and there

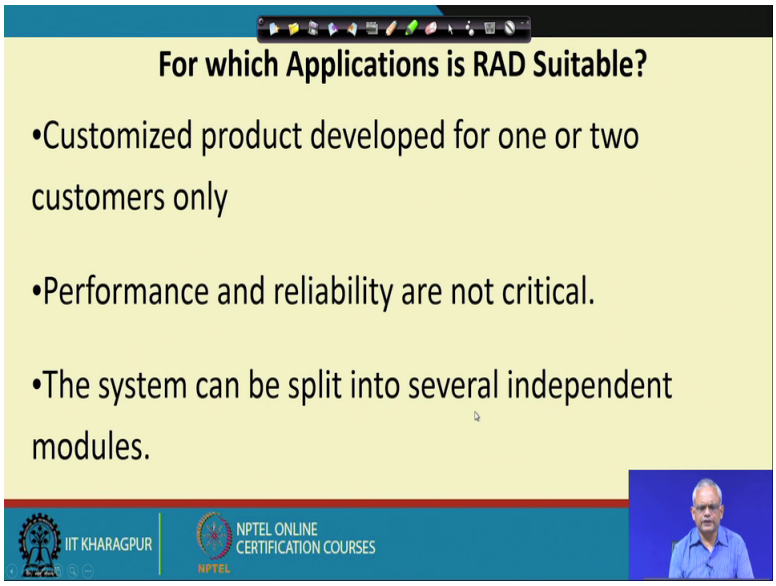
is the customer gives the feedback, this prototype itself is refined based on the customer feedback.

So, this is a very different style advocated by the RAD development; for rapid development, the prototype is refined into the actual software. Remember that in the prototyping model, the incremental the prototyping model the prototype was a throwaway software. And of course, the incremental and the evolutionary model, each increment was not a prototype, but a properly planned developed according to some lifecycle may be the waterfall model.

(Refer Slide Time: 17:37)

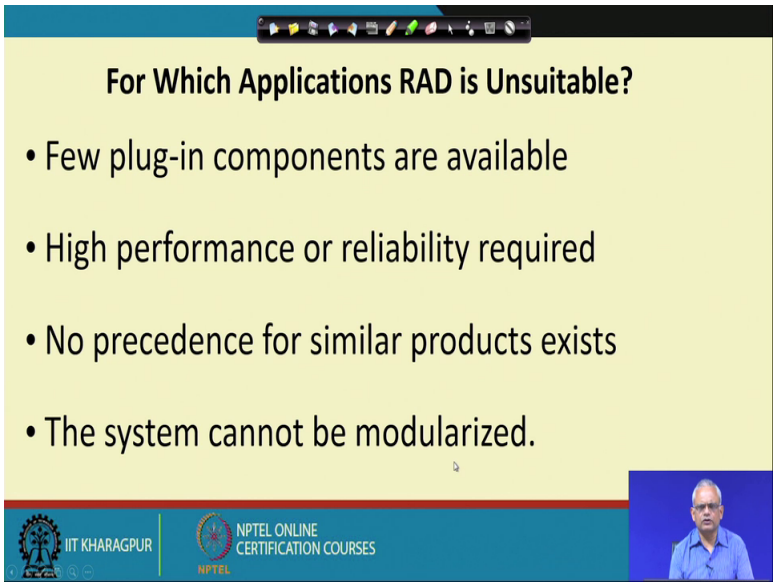
As the emphasis of the RAD is to facilitate faster development, it creates prototype using specialized tools and the development can be fast by using visual style of development, drag and drop style, use of reusable components and also standard APIs. Standard APIs need to be used because that helps in the use.

(Refer Slide Time: 18:16)



The RAD model is suitable for software that is developed for one or two customers that is customization. And here, since the design is not so structured; the code quality is not so great. The project should not have performance and reliability at a premium and also not suitable for very very small software. Software should be able to split into several independent modules.

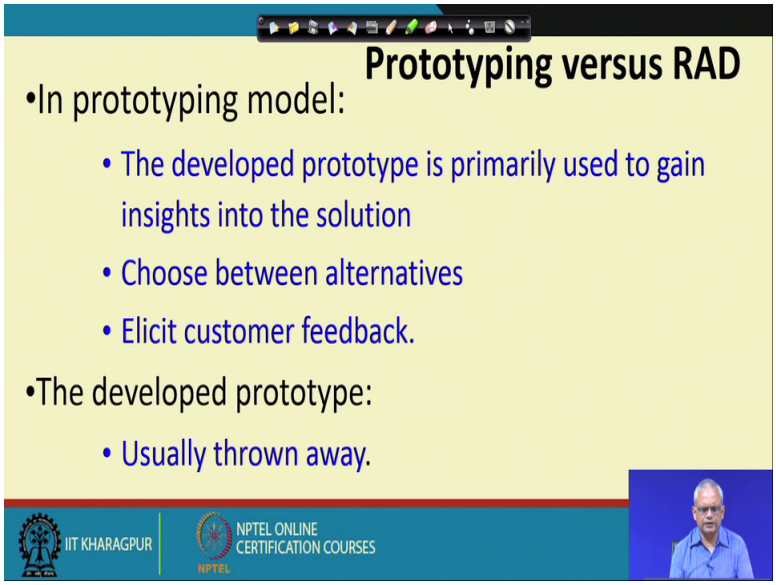
(Refer Slide Time: 18:58)



And also for the RAD model to really work, we should have many reusable components.

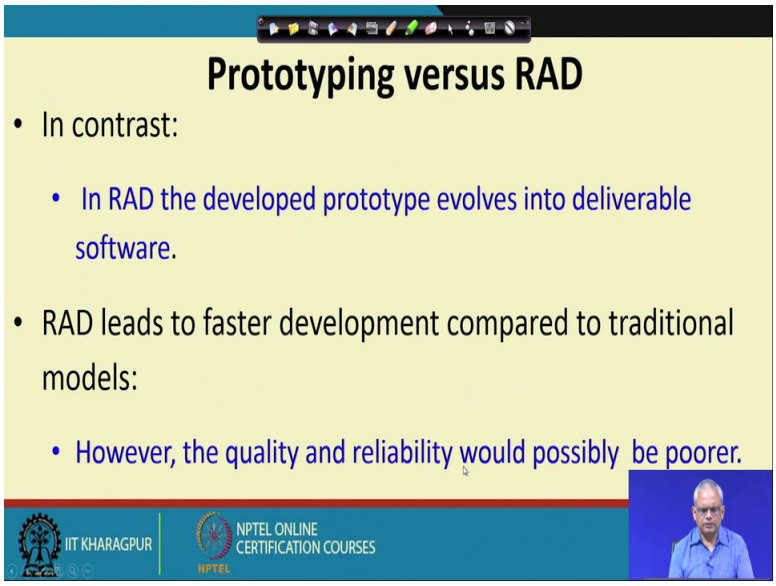
So, if reusable components that is a new product where reusable software is not available, then run RAD model will be unsuitable. If we need high performance or reliability, RAD model is unsuitable. There are no precedence for similar products, then again RAD model is not useful and also the software should be reasonably large which can be modulized into several modules.

(Refer Slide Time: 19:48)



It is easy to compare prototyping with the RAD model. In the prototyping model the developed prototype is a throwaway prototype. The main reason the prototype is developed is to get the customers suggestions and also insight into the solution.

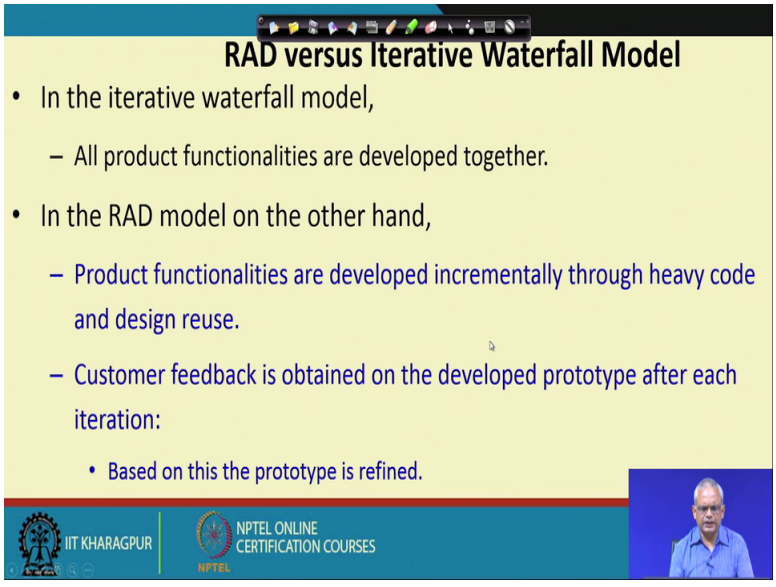
(Refer Slide Time: 20:03)



The developers can choose between alternatives by developing the prototype and once the prototype has been used to evaluate alternatives, client feedback is obtained the prototype is thrown away and new development completely planned development starts in a iterative waterfall model.

In contrast, in a RAD model the prototype itself is modified into deliverable software. Of course, that matches with the requirement or the objective of the RAD which is faster development, but quality and reliability become a question.

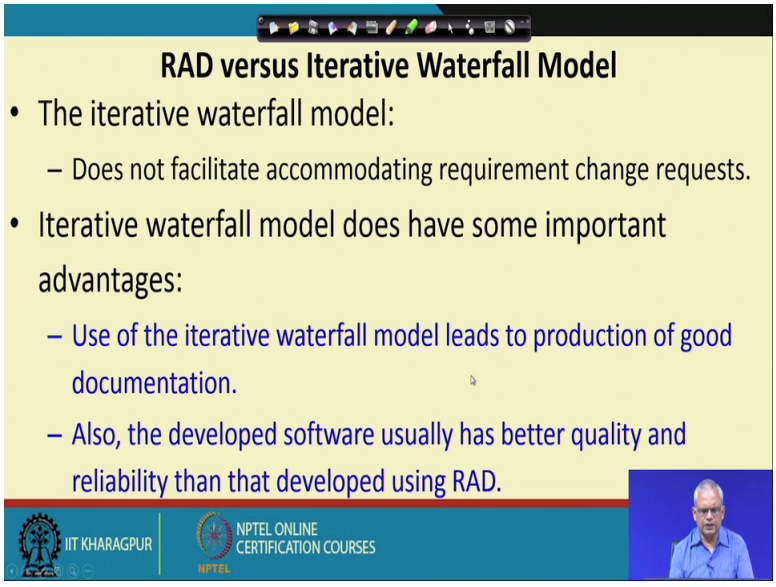
(Refer Slide Time: 21:00)



But how does RAD compared with iterative waterfall model?

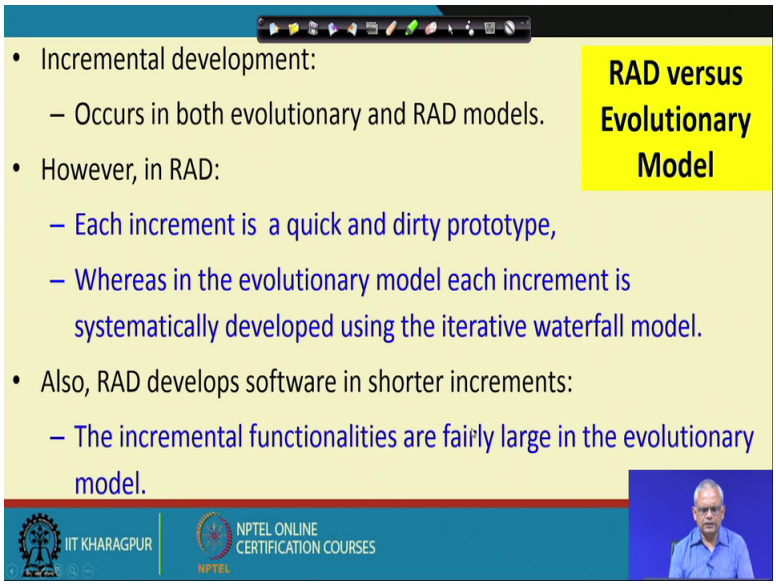
Iterative waterfall model, initially all requirements are captured all functionalities are developed together, whereas in RAD model these are incrementally developed. So, it has similarity with the incremental development model and on each the customers feedback is obtained. And therefore, client interactions customer interactions are high here.

(Refer Slide Time: 21:29)



It is easy to change to accommodate requirement changes based on customer feedback. But then, compared to RAD model, iterative waterfall model usually results in good quality design higher reliability and also production of good documentation.

(Refer Slide Time: 21:57)

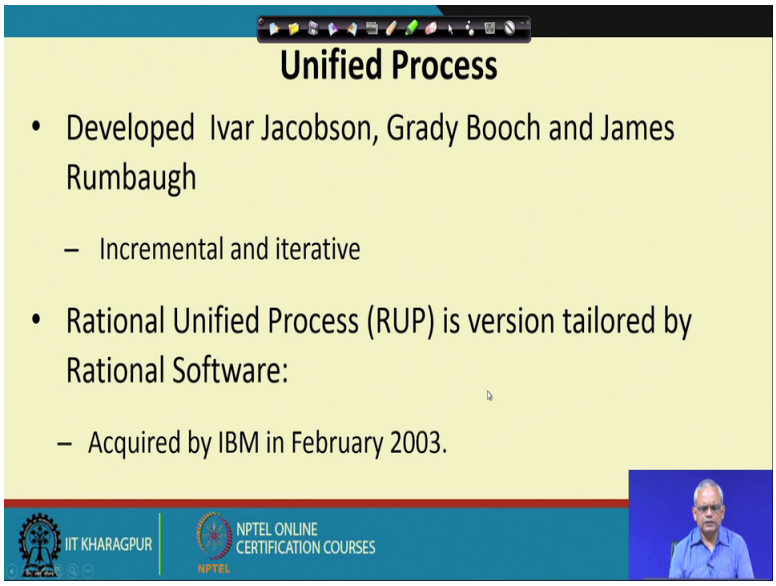


How does RAD model compare with the evolutionary model? In both RAD and evolutionary increments are deployed the development occurs over several increments. In RAD the increment is a quick and dirty prototype. On the other hand, in the evolutionary model each increment is developed using iterative waterfall model.

And if we look at the size of the increment, the RAD increment are shorter called as time box and on the other hand, the in the evolutionary model each increment takes longer time and more functionality is completed in the iteration.

Now, let us look at another very popular process the Unified Process.

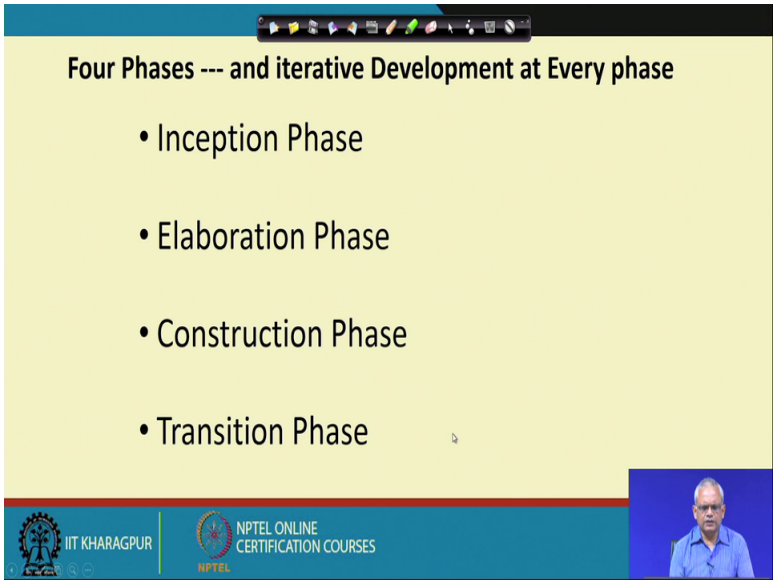
(Refer Slide Time: 23:03)



This was developed by Jacobson, Booch and Rumbaugh. Extensively used for object oriented software development and it is incremental and iterative. So, just note the word incremental and iterative, it means that the features had to be identified beforehand. And then, over several iterative development the software is developed each time deployed at the client side client feedback is obtained.

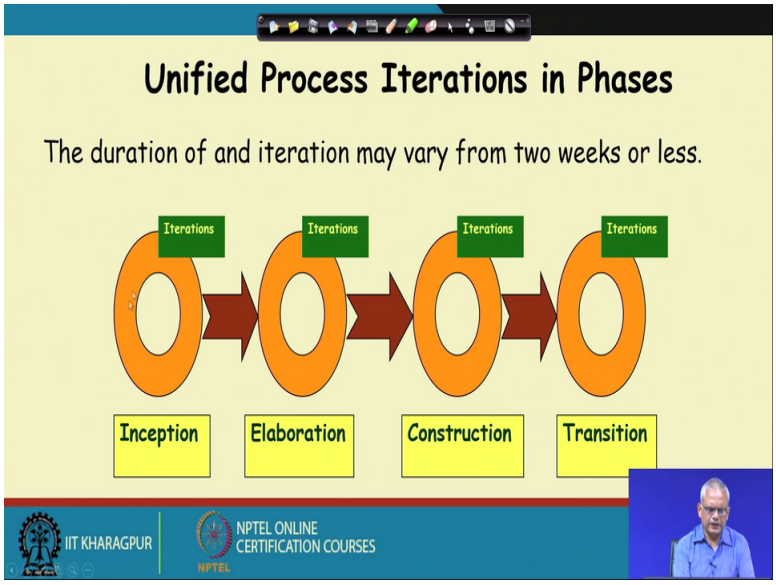
The unified process tailored by the rational corporation is called as a rational unified process and of course, the rational corporation was acquired by IBM in 2003.

(Refer Slide Time: 24:01)



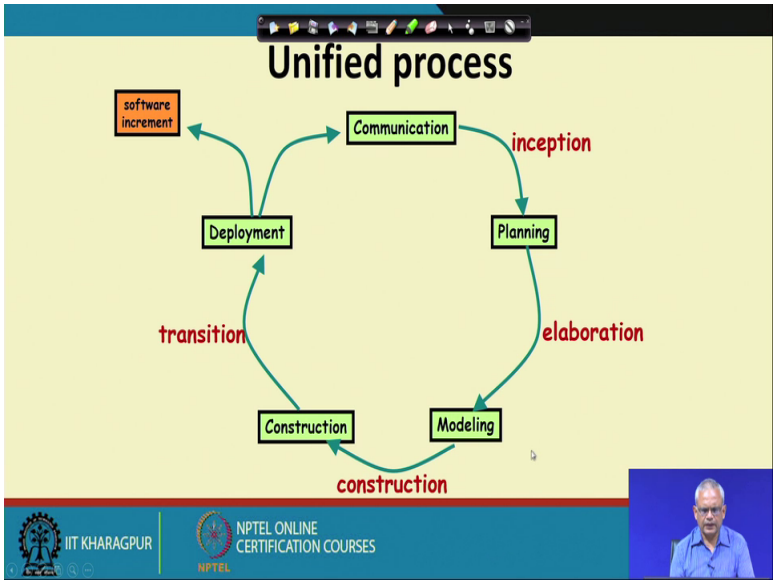
The development here in the unified process occurs over 4 phases; Inception, Elaboration, Construction and Transition. In each of these phases, consists of many increments.

(Refer Slide Time: 24:18)



The inception phase occurs over many increments or iterations; the Elaboration also many iterations can be there; Construction several iterations and finally, the transition. Let us look at what these phases imply Inception, Elaboration, Construction and Transition?

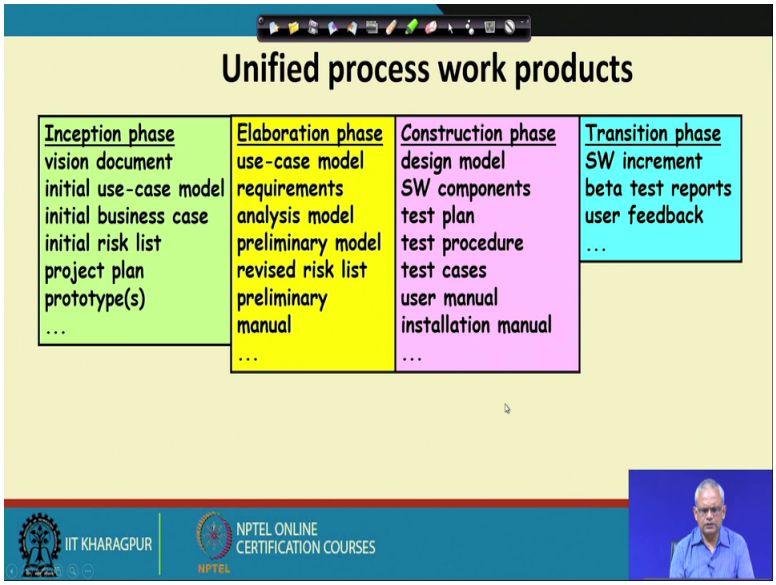
(Refer Slide Time: 24:44)



In the Inception phase, communication with the customer and the features are identified and also plan about how to develop and during the Elaboration, the features identified features are modeled. And finally, the Construction the software is constructed is developed and during Transition the software is deployed in the client side.

But then it is not as simple as that.

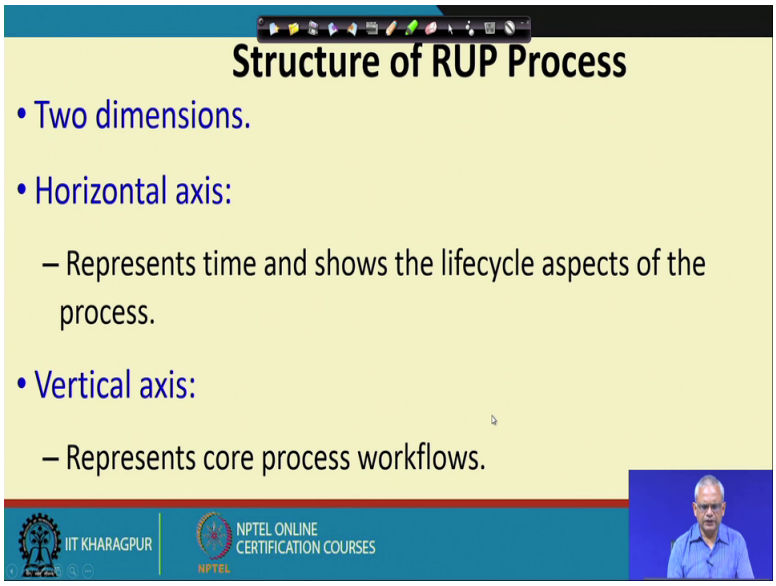
(Refer Slide Time: 25:26)



We will see what exactly occurs that was a much simpler model to understand. Now, let us see what are the output of every phase? In the inception phase, the initial use case model that is the features, the business case, risk list, project plan. So, these are the planning documents and any prototype.

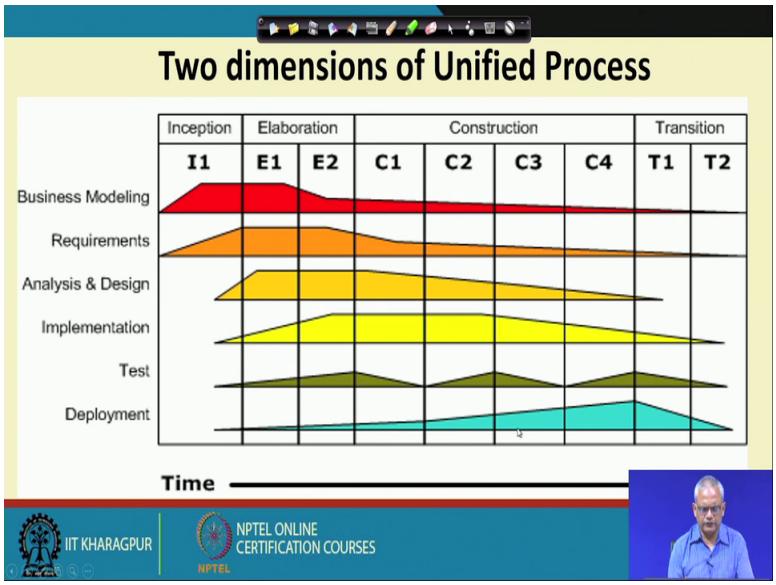
In the elaboration phase, models are made analysis model preliminary model, manual use case model and so on. The construction phase, the software is developed; the test plan is made, several manuals, user manual and installation manual are written and finally, these are deployed at the customer site; the beta test reports and the user feedback is obtained.

(Refer Slide Time: 26:29)



But then if we want to visualize this graphically which will show a more realistic picture the rational unified process. The horizontal axis will show the lifecycle aspects and the vertical axis will represent the core process flow.

(Refer Slide Time: 26:47)



Just see here the four phases are marked here and each one occurs over several iterations. The requirements and the planning peak during the inception phase. And even they continue during the liberation and construction phase. The analysis and design, they peak during the elaboration and construction and slowly taper off. Implementation peaks

during the construction phase and tapers off; but then, there are some construction during the elaboration phase also and test is present throughout and deployment peaks towards the transition.

We are nearing the time for this lecture. We will stop here and continue from this point in the next lecture.

Thank you.