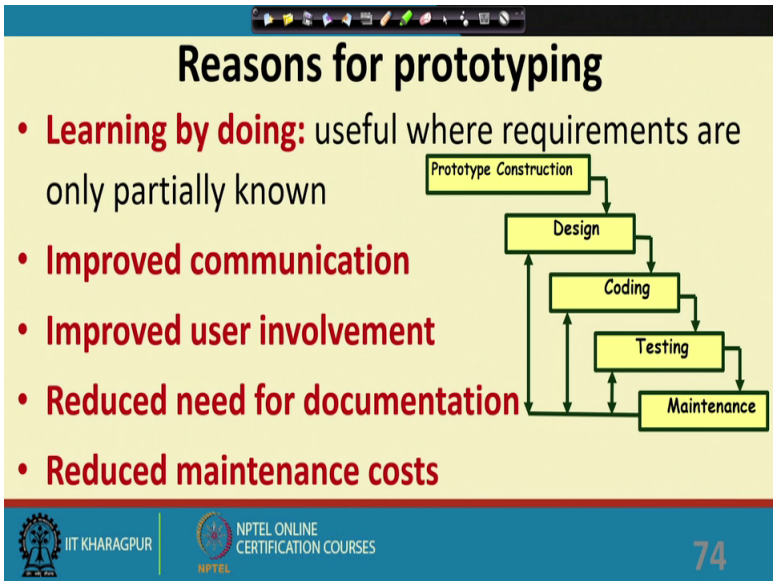
Software Engineering
Prof. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 10
Incremental Model

Welcome to this lecture. In the last lecture, we were discussing about the Prototyping model. The prototyping model is a variant of the waterfall model.

(Refer Slide Time: 00:32)



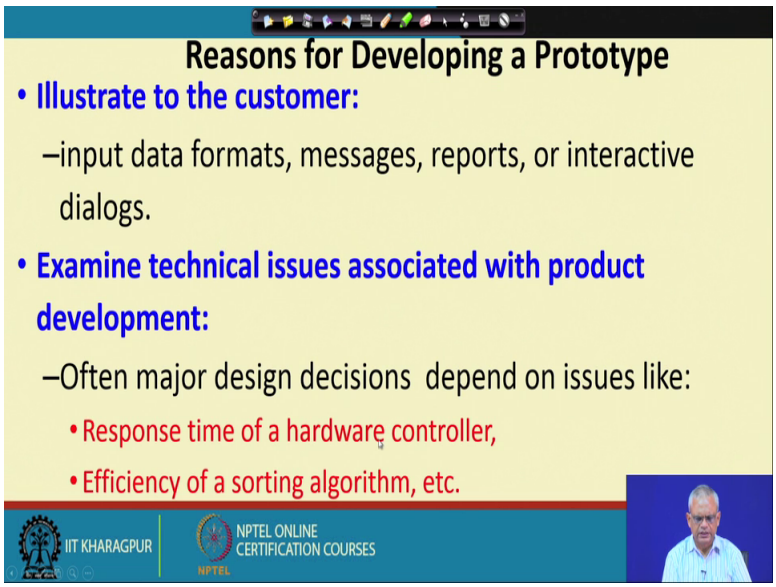
And we said that in the prototyping model in place of the requirements analysis and specification we have the prototype construction. First a prototype is constructed and then it is demonstrated to the user and any changes that are requested by the customer are accommodated at this point, and once the customer agrees to the prototype, then the development starts using the waterfall model. There are many advantages of the prototype model. First is that when the requirements are written, it is not really known, it is very difficult to identify what is missing what needs to be changed which are ambiguous and so on.

But if the developer start developing the prototype, then they become clear that what are the difficulties with the requirements. It improves the communication with the customer rather than just exchanging documents or just talking in different meetings, here they can see something real life and they can provide their feedback on that.

The user feels involved in the development because his comments you can see that the product is changed as per his comments. There is a reduced need for documentation because the prototype serves the purpose of a requirement. A service document is not there actually because the prototype serves as a animated requirement specification; of course, if necessary depending on the project and SRS document can be written based on the prototype.

But then, normally a SRS document is not needed because the prototype serves as the requirement specification. It also results in reduced maintenance costs because the development quality is good. The developers first developed the prototype the gain expertise by doing a prototype and then, they are better prepared to carry out the actual development and normally the software has better quality. And therefore, the maintenance costs are lower.

(Refer Slide Time: 03:28)

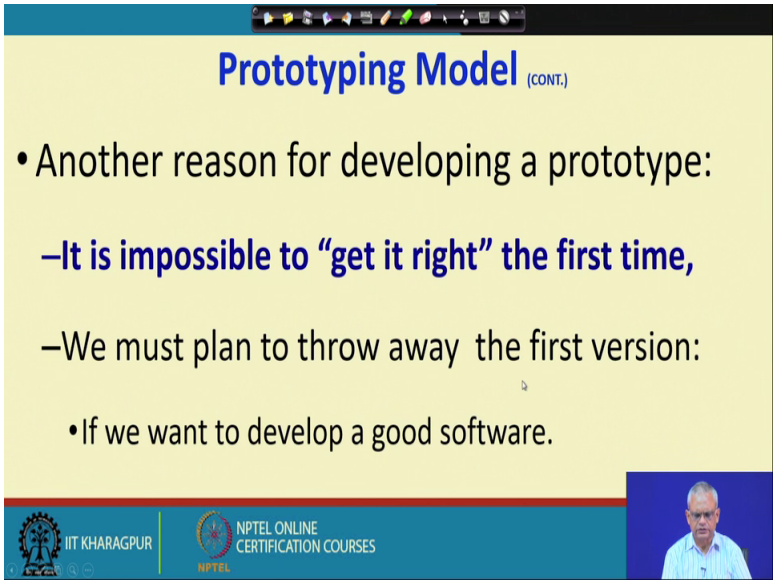


But then there are 2 main reasons, if we have to summarize what are the main reasons why the prototype model can be used, there are 2 reasons. One is that the customer can be illustrated about the software. Especially they can look at the input data formats; the messages being displayed, reports being produced, interactive dialogues, because this user interface part is to a large extent is a personal choice. A customer if he is given a user interface he would like some aspects to change to meet his choice.

And that is the reason why for almost every project the GUI part is developed using a prototype model and even for some software, the entire development is according to a prototype model. But the GUI for every project typically is through a prototype model where a quick prototype is made for the GUI and the client feedback is obtained. It also allows the developers to examine the technical issues for which they are not sure.

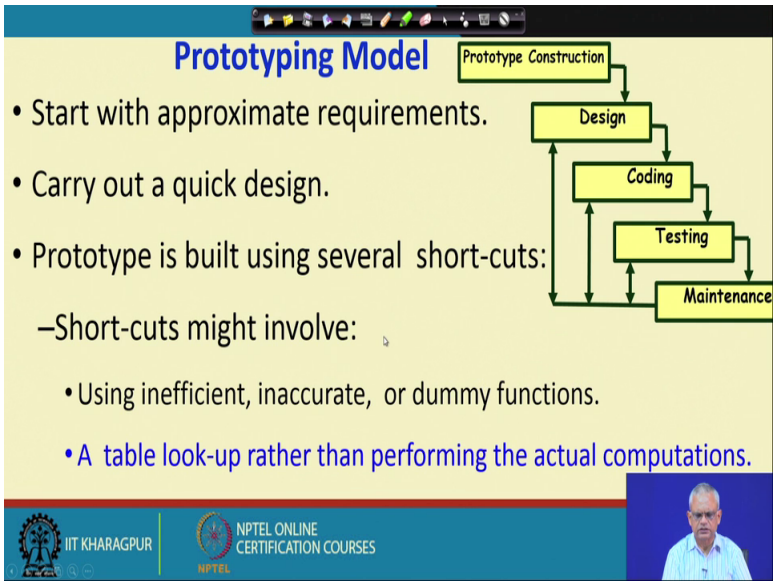
For example, the design decisions might depend on what is the response time of a hardware controller or maybe what is the setting time of some data. And if they can experiment that using a prototype they can design it better.

(Refer Slide Time: 05:29)



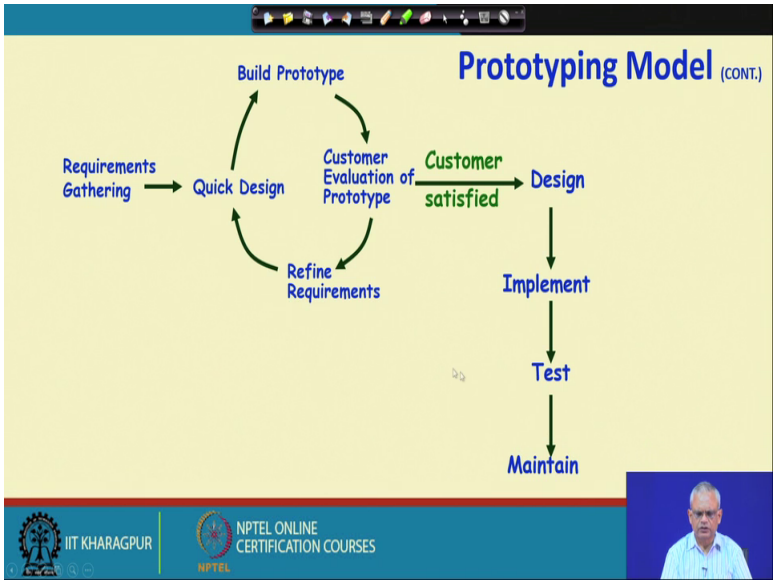
There is another reason that it leads to a good quality software. It is impossible to get it right the first time. The first development is actually a lot of learning rework and so on and based on the experience, if the actual development starts. Then, it becomes easy and also good quality of software can be obtained and that is the reason why many experienced developers say that must plan to throw away the first version: if we want to develop a good software.

(Refer Slide Time: 06:15)



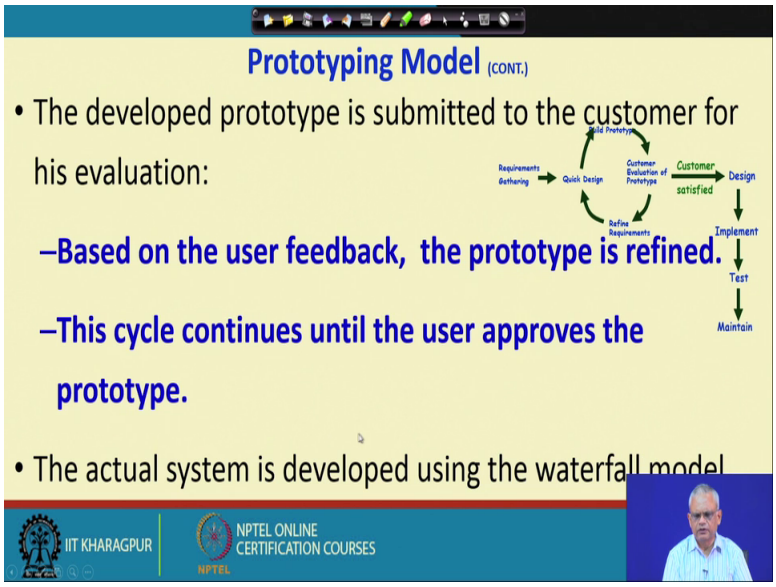
Now, let us look at the activities that take place here. Starting with approximate requirements from the customer the prototype is built. It is demonstrated to the customer quick design and then, demonstrated to the customer and the prototype is built using several shortcuts maybe dummy functions, table look up rather than the actual computation and so on. And the client feedback is obtained on the prototype. And as long as the client agrees the actual development starts.

(Refer Slide Time: 07:02)



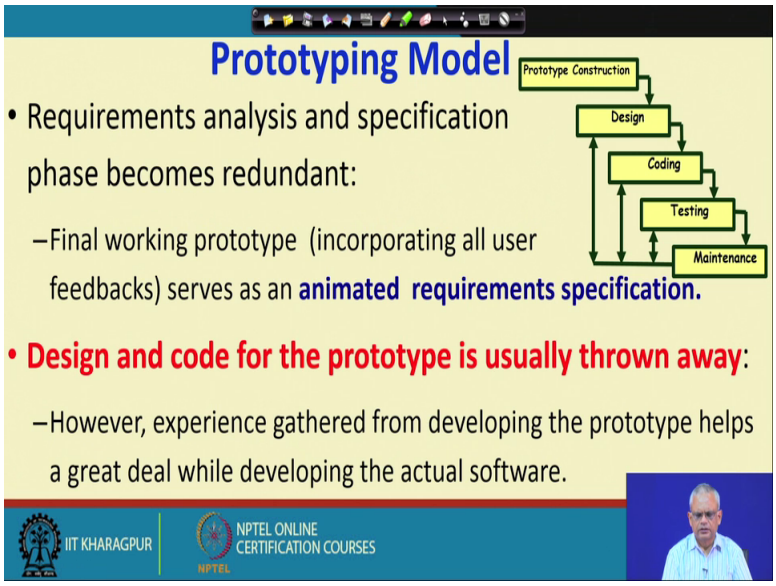
We can represent the prototyping model with this schematic. Initial requirement gathering there is a quick design build prototype, give the prototype to the customer for evaluation, and based on the feedback refine the requirements. Again do a quick design based on the changed requirements, refine the prototype and so on. It goes on this cycle until the customer is satisfied and then, the traditional waterfall model of development starts.

(Refer Slide Time: 07:38)



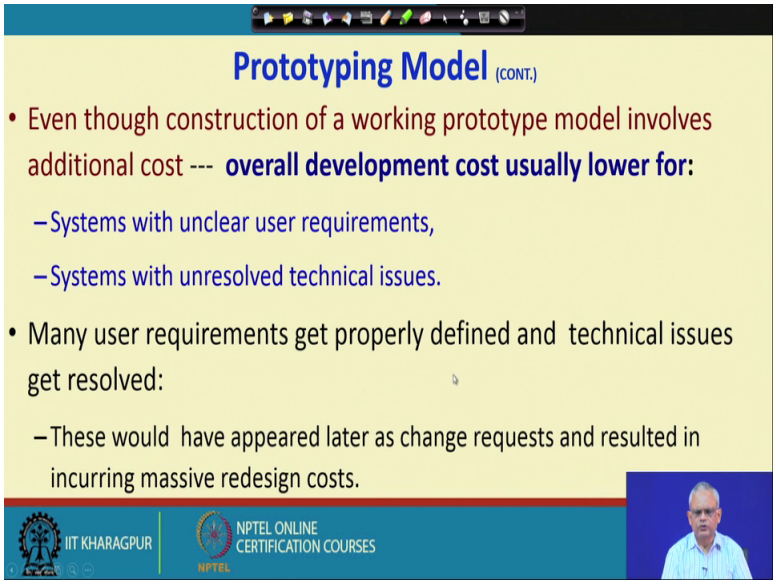
So, the actual development here is the waterfall model and the main difference with the waterfall model is the prototype construction.

(Refer Slide Time: 07:59)



The prototype serves as animated requirements specification. Often the SRS document is not needed. The prototype serves for the SRS document. But it is important to remember here is that the prototype once it works, it is not really refined into the actual software. The software is written from scratch again. The initial prototype is thrown away; new software is written. But then, the experience gathered from developing the prototype helps the developers.

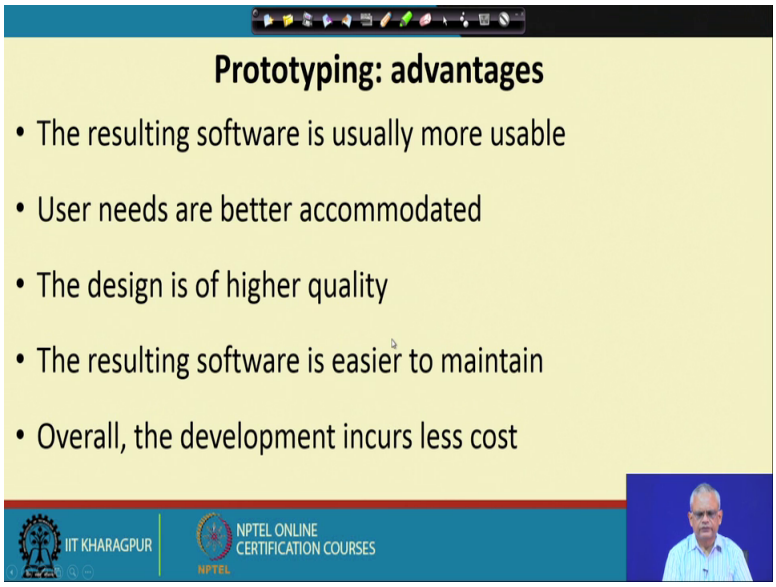
(Refer Slide Time: 08:44)



It might appear that the prototype development requires additional cost and therefore, the development would be costly, not really in many cases the overall development cost is lower especially when the client is a unclear about the requirements. Then, later there will be too many changes and also when the technical issues are not clear to the development team. Starting the development would finally, cost more rather than experimenting upfront by developing a prototype that would be a better idea.

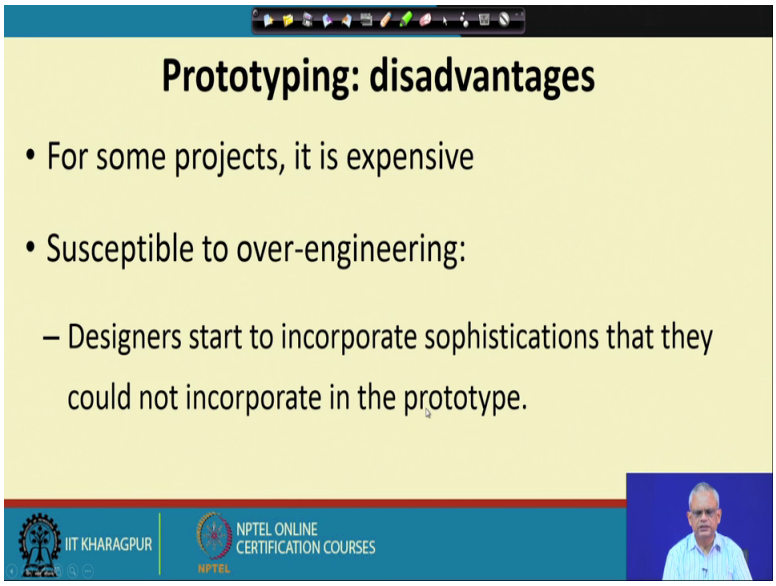
So, for many types of projects, prototype construction may be a small overhead, but lot of saving can be there in actual development. Many user requirements get defined technical issues get resolved and therefore, change requests are reduced to a large extent and results in a good quality software. And therefore, the prototype model is often preferred and especially for the GUI the prototyping model is normally used.

(Refer Slide Time: 10:16)



The resulting software is more usable. User needs are better accommodated. Design of good quality; the software is of good quality. The resulting software easier to maintain and the overall, development cost may be low.

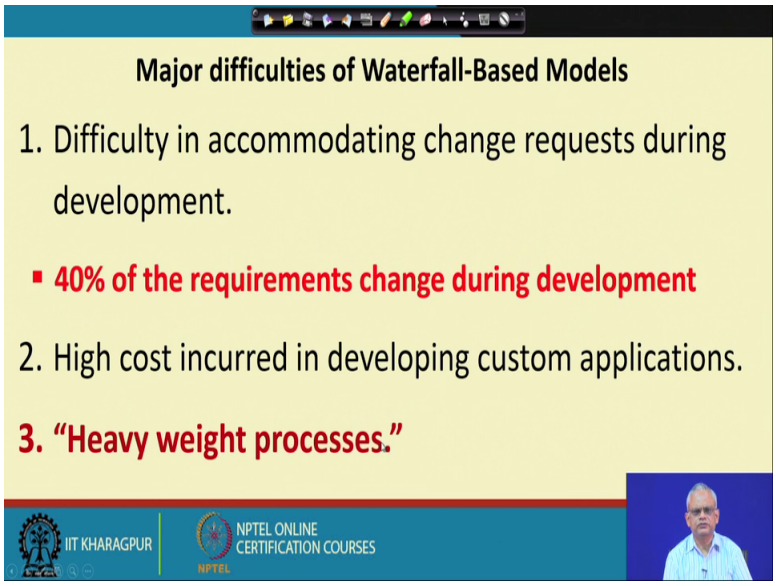
(Refer Slide Time: 10:39)



But then, what are the disadvantages? For some projects prototyping may be expensive. For example, if all the issues are understood the requirements are clear, then the prototype construction will just add to an extra overhead. Another difficulty with the prototype model is over-engineering.

Once, the developers develop the prototype, then they start to incorporated sophistications that did not incorporate into prototype and even if some features are not required, they might be tempted to do it, because they had experience in writing simple prototype; they would like to make it much more sophisticated. So, when the prototyping model is used, it is often the case that developers feel tempted to over engineer the actual software.

(Refer Slide Time: 11:47)



We looked at the waterfall model and also some popular variants of the waterfall model namely the V model and the prototyping model. Now, let us see the major difficulties with the waterfall based model and why these models were later not used so much and newer models came up. One of the big problem is that the requirements need to be frozen, before the project development starts; that is the characteristic of every waterfall based model.

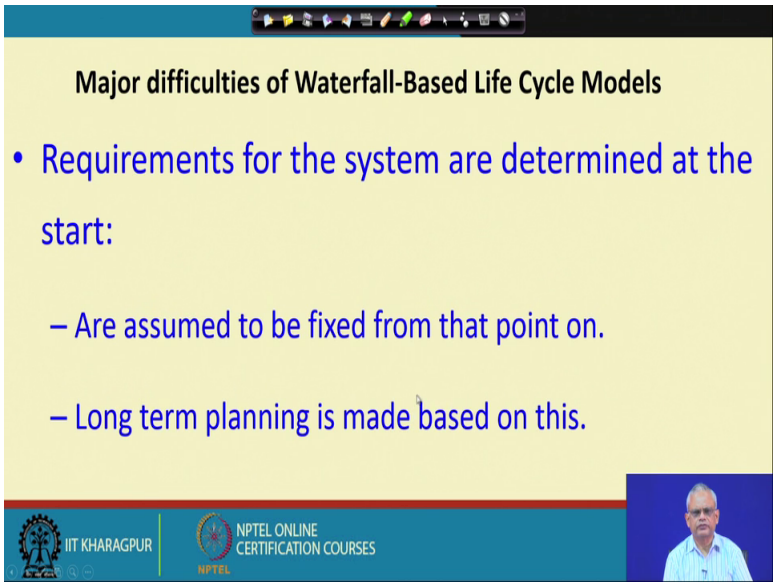
But in reality, it is observed that about 40 percent of the requirements change as the development starts and in this situation, if a waterfall model is used. Then the final developed software will not meet the user requirements. They will be unhappy and lot of changes they would suggest at the end. We are saying that now the customized software or the service software are increasing in a large way and the waterfall model is good for software that is written from scratch edge was being done in 1970's and 80's.

But now, in the customization of a applications using waterfall model becomes problematic, because the waterfall model need to define the requirements all the requirements; then design code test and so on. The customization maybe just we need to change some small aspect of a already working software. So, the work here may involve understanding what is required to be changed changing only those part doing a regression test and so on and those are difficult to accommodate within a waterfall model.

The last important difficulty with a waterfall based model is that these are "Heavy weight processes." We call the waterfall based models heavy weight processes because lot of documentation is produced. At the end of every phase documentation is produced which are reviewed and a symptom of all waterfall based models is that at the end of the software development.

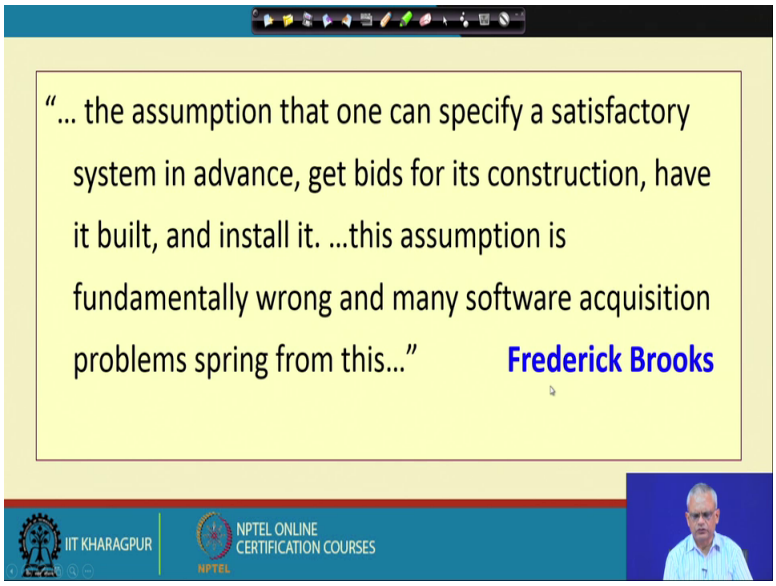
There is a huge mountain of documents and that is the reason why the waterfall model is called heavy weight processes. It is observed that in waterfall model nearly half the time, the developers are documenting and documenting is not real development. So, is it possible that if we reduce the documentation, the development can be done faster? And we will see that in the later development models namely the Agile models the documentation is kept to minimum.

(Refer Slide Time: 15:48)



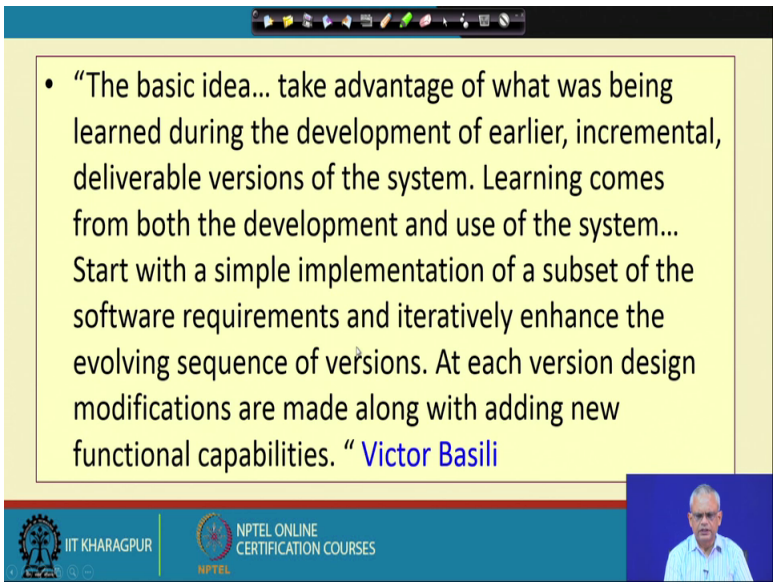
But one of the major difficulty, I will say that this is one of the difficulties which is acutely felt is that requirements need to be defined at the start. And from that point it becomes fixed and not only that the project manager makes plans based on the requirements that what are the phases, how much time the phases will take and so on. And any change their upsets the manager's plans and it increases the cost to a large extent if any change is accommodated.

(Refer Slide Time: 16:35)



Let us see what Frederick Brooks, a celebrated authority in this area has to say "the assumption that one can specify a satisfactory system in advance, get bids for it is construction, have it built, and install it. This assumption is fundamentally wrong and many software accusation problems spring from this" Brooks has experienced in many projects and he observed that for any customer becomes very difficult to give all the requirements in advance and then, get somebody to construct it. He says that this is a fundamentally wrong and most project failures can be attributed to this.

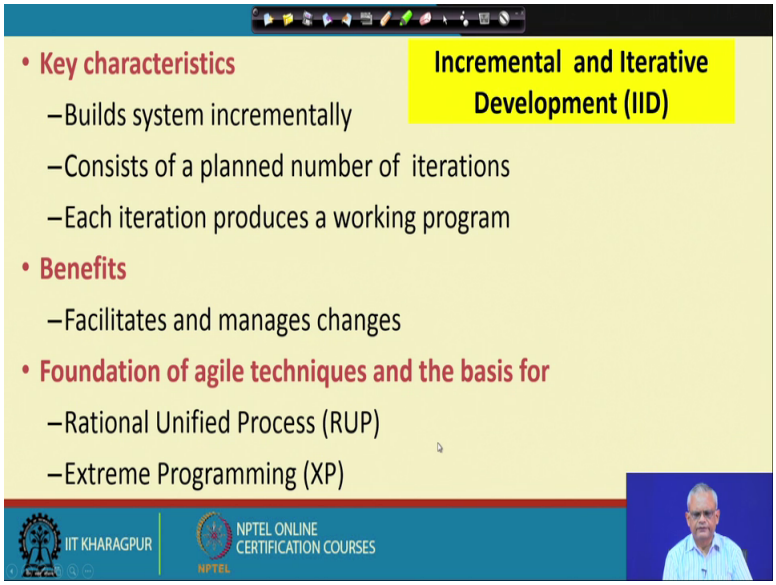
(Refer Slide Time: 17:32)



Let us look at the Incremental model. In the incremental model, we develop a increment and then we learn by developing one increment and then, do the next increment and make we have the increment installed at the client place and then who take the feedback and in the next increment, we accommodate it and this is iterative enhancement of the versions.

So, let me just read it. "The basic idea, take advantage of what is being learned during development of a increment. Learning comes from both development and use of the system. Start with a simple implementation of a subset of the requirement; iteratively enhance the evolving sequence of versions. At each version design modifications are made along with adding new functional capabilities."

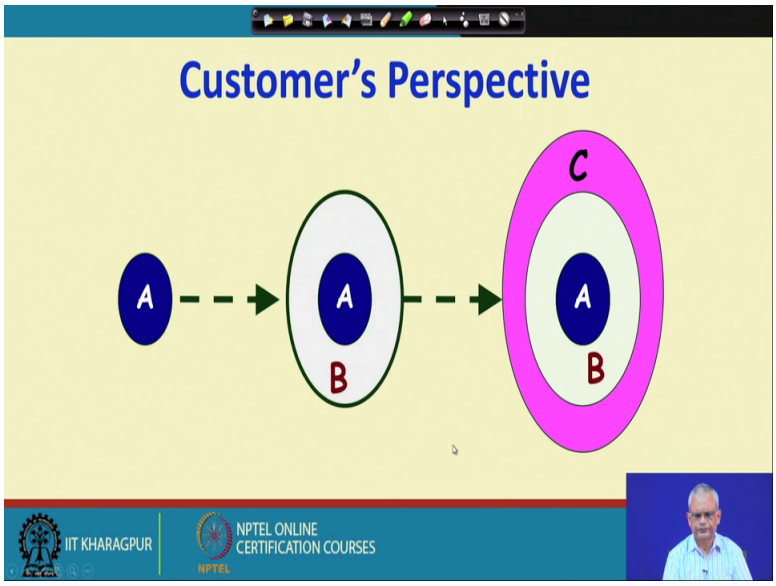
(Refer Slide Time: 18:36)



The key characteristic of the incremental and iterative development is that builds the system incrementally. There is a planned number of iterations. So, the initial requirements are split into versions and those are the increments. Each iteration; here produces a working program which is installed at the clients place.

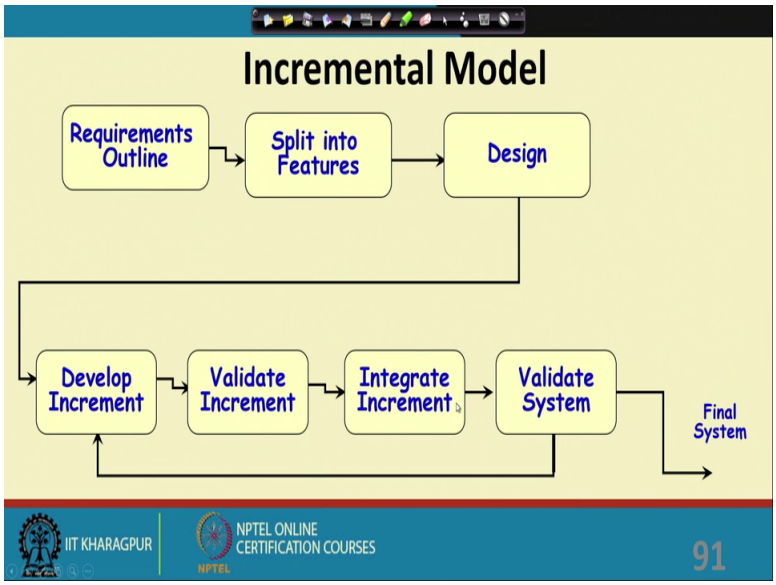
The main benefits are that the client can give feedback which can be incorporated in the next increment and these techniques are actually the foundation for the agile techniques. The rational unified process and the extreme programming the incremental development has many advantages.

(Refer Slide Time: 19:38)



If we look at the customers perspective of a incremental development, initially gets a core part of the software, experiments gives feedback. Then he gets a slightly larger software and so on, until he gets the full software.

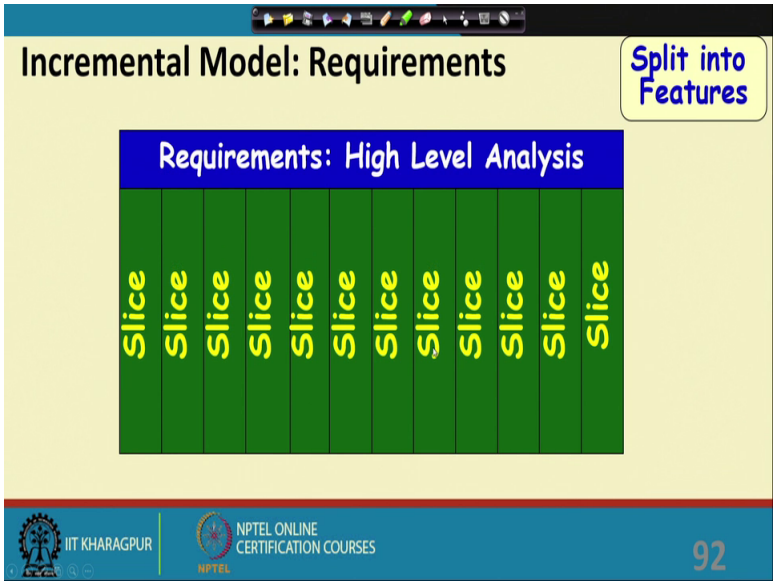
(Refer Slide Time: 19:59)



But as far as the designers are concerned, the developers the model they follow is something like this that they get the requirements, initial requirements and they split this into deliverable features. And they do a overall design and then they keep on developing the increments, test the increment, integrate with the previous software, validate the

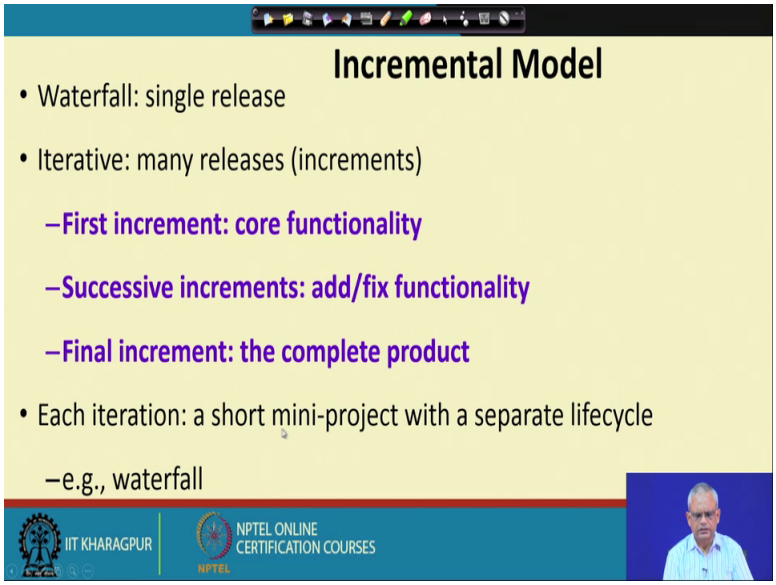
system and keep on doing this until all the features are accommodated. And then finally, they deliver the system.

(Refer Slide Time: 20:46)



The initial requirements are split into features and they then they go on developing one feature at a time.

(Refer Slide Time: 21:01)

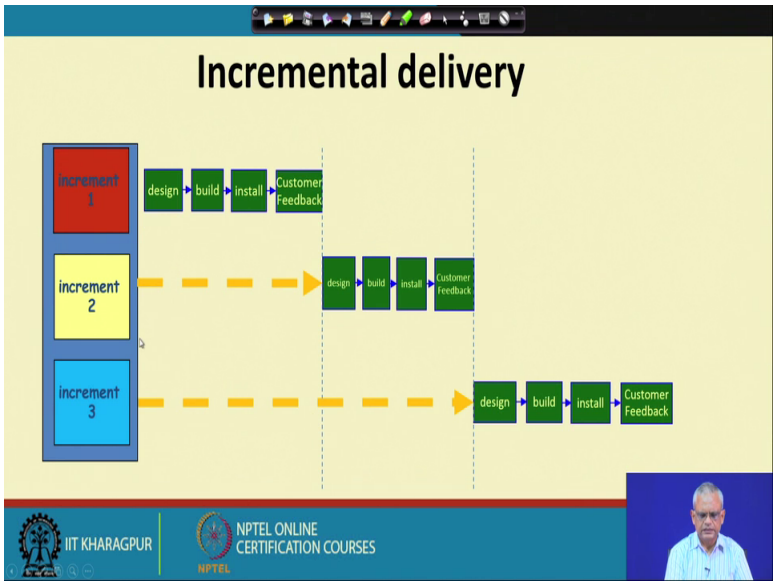


If we compare the incremental model with the waterfall model, the waterfall model has a single release; but in iterative model there are many religious to the customer. The first increment is typically, the core functionality and on each increment, new functionalities

are added and some parts that are the customer provides feedback are modified and the final increment is the complete product.

But then, each iteration is a short mini project which may be used in a life cycle like waterfall.

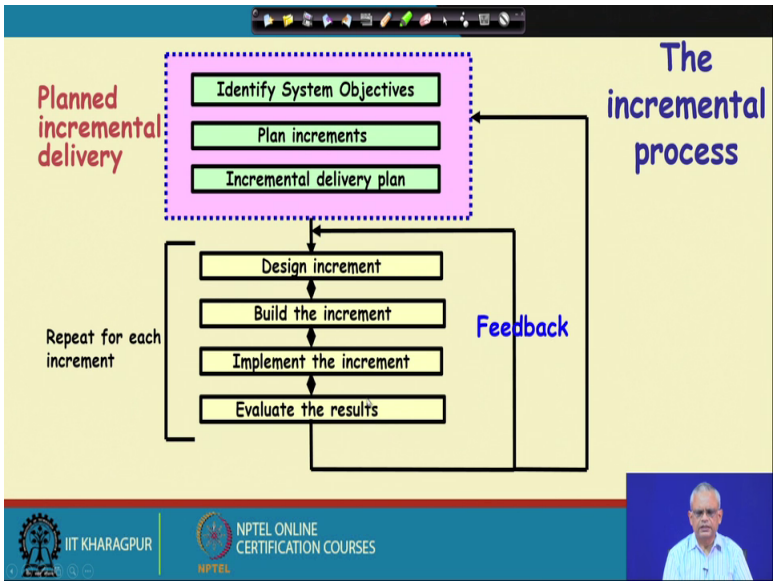
(Refer Slide Time: 21:54)



So, if we see here the entire requirement is split into increments; the first increment design build install and customer feedback. And once the first increment customer feedback is obtained, the second increment has started and then, the third increment is started and so on.

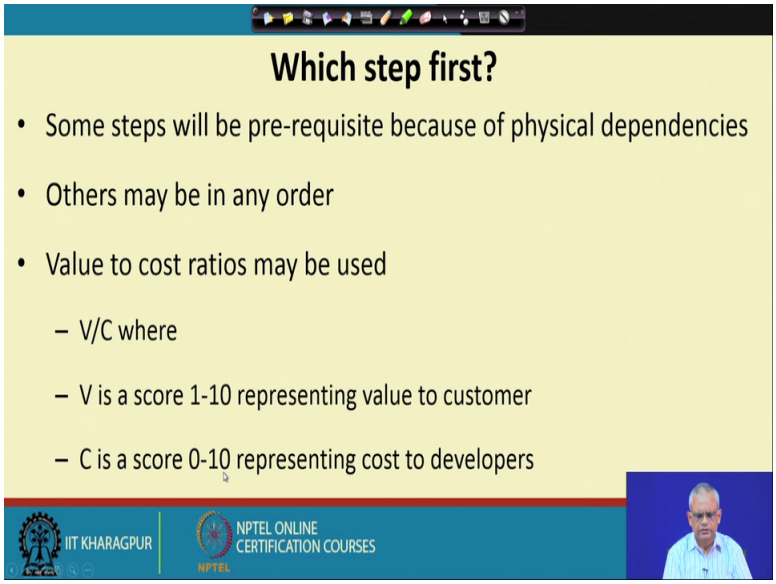
And this in this time to develop one increment is typically called as a time box.

(Refer Slide Time: 22:34)



We can represent the incremental model initial objectives or our system objectives obtained. The increments are planned the incremental delivery when which will be developed is planned. So, these are the initial requirements and planned for the delivery. And then, the development starts each time one increment is designed build the increment implement and evaluate the results and goes on building one increment after other until the full software is developed.

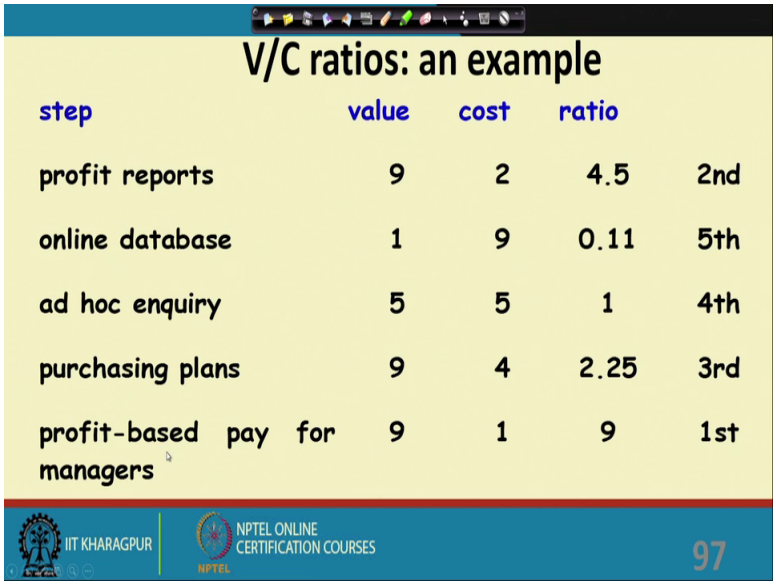
(Refer Slide Time: 23:17)



But then, one question is that once the initial requirements are identified split into several increments. There may be n number of increments let us say 10 increments are planned.

But which increment will be done first? How does the developers decide which increment to be taken off first, let us look at that. One thing is that some increments have to be done because the other ones will depend on that. We have to do that first because the others might have dependency on that. But then, if some requirements are can be done in any order, then we can compute a value to cost ratio. This is called as the V by C ratio; where, V is the value to the customer, we can give a value you can ask the customer to give a value 1 to 10 and the cost is the cost of development can I give a value 1 to 10.

(Refer Slide Time: 24:40)



This is just a example here that these are the features. These are the values that the customer specifies how valuable are the features and this is the cost of the development. The development team gives a cost and then, the ratio is worked out and you can see here that this is the largest V by C ratio and we say that this will be the first increment.

We will next discuss about the evolutionary model with the iterations. We saw that incremental model is a very important model, it has lot of advantages and next we look at the evolutionary model and then based on these two models we look at the agile models.

Thank you.