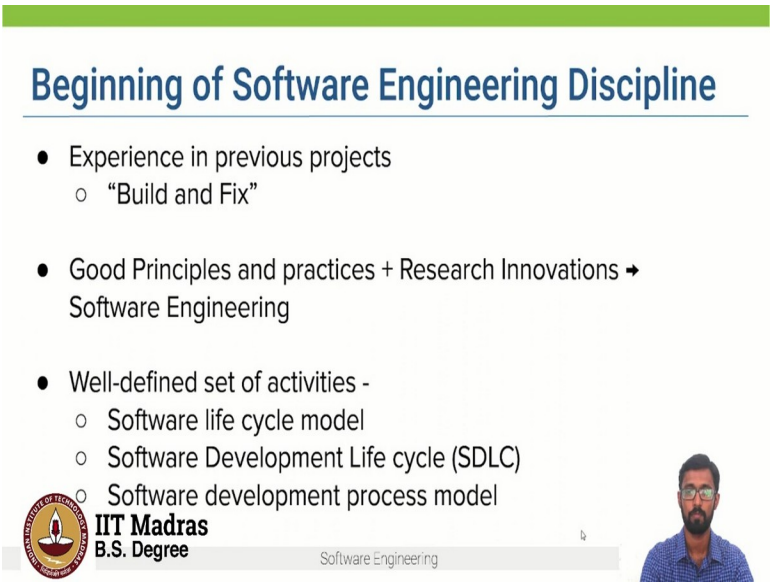
Software Engineering
Professor Doctor Sridhar Iyer
Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Doctor Prajish Prasad
FLAME University
Software Development Models - Plan and Document Perspective

(Refer Slide Time: 0:24)



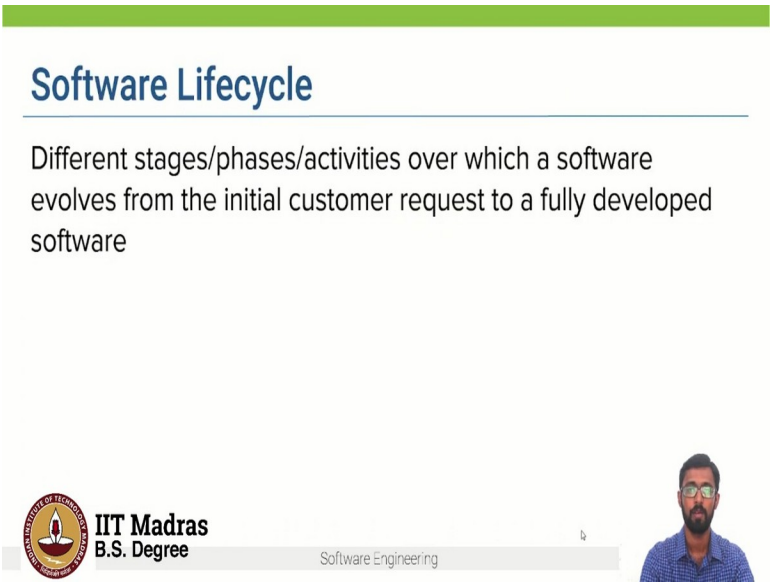
In the previous lectures of this week, we looked at various phases and activities in the software development process. We also looked at why these phases are important. So, learners many of you might have developed software projects before. So, let us reflect on how you went about developing software.

You might not have strictly followed the process of identifying requirements, then coming up with the design, then writing code testing it and so on. So, you might have followed an exploratory programming style. So, this is commonly known as the build and fix type of programming.

And here programmers, they typically start to write the program immediately after forming an informal understanding of the requirements. And once the program is complete, then they fix anything which does not match the users expectation. And this was how programming was done in the early 1950s and 60s. So, good programmers, they knew certain principles that helped them write good programs. And these principles could be things like thinking about the big picture view of the system, which is the design before you actually start writing the program.

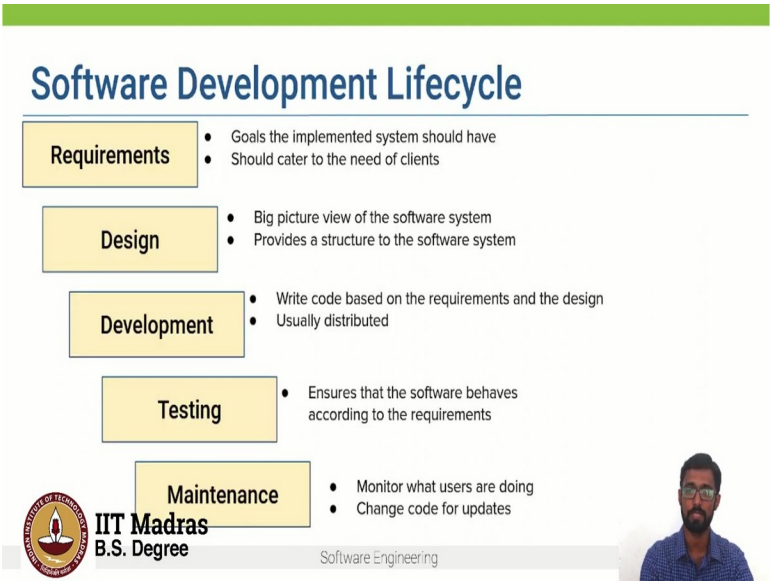
And over time, these good principles and practices along with research innovations, they have been systematically organized into what is known as software engineering today. So, software Engineering emphasizes software development to a well defined and ordered set of activities. And these activities are commonly referred by various names such as the software lifecycle model, software development lifecycle or SDLC and the software development process model.

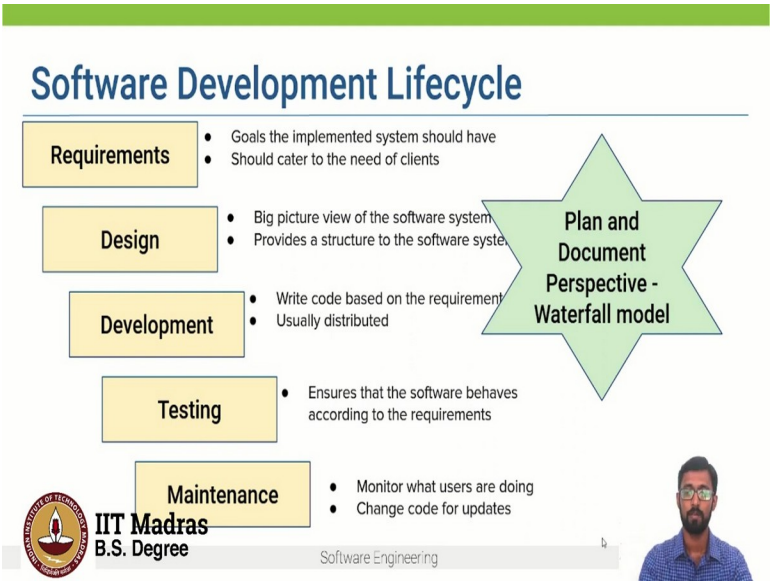
(Refer Slide Time: 2:36)



So, the software lifecycle is the different stages, the phases and activities over which is software evolves from the initial customer request to a fully developed software. And we saw some of these activities in the previous lecture.

(Refer Slide Time: 2:54)

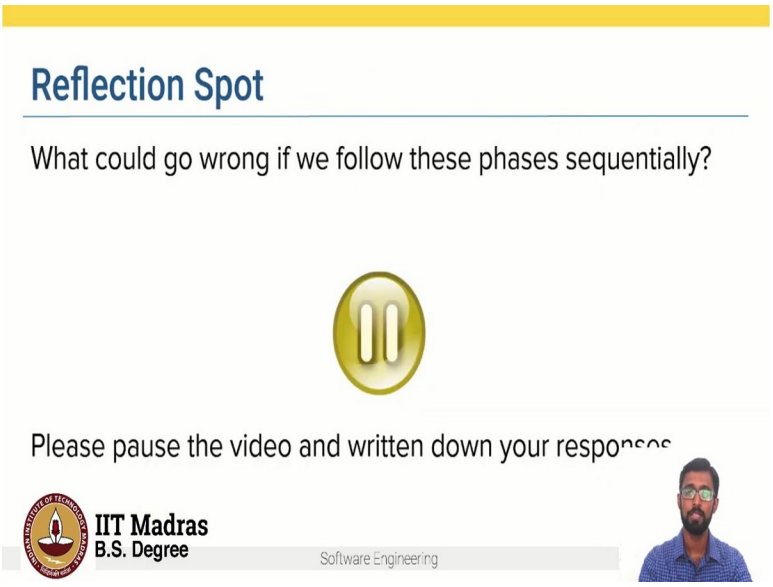




So, we start the development by gathering requirements, which should cater to the need of the clients. Then we come up with a design which is a big picture view of the system. Then developers write code based on the requirements and the design. And then we do testing and maintenance. And this process is commonly known as the waterfall model, or the Plan and Document perspective.

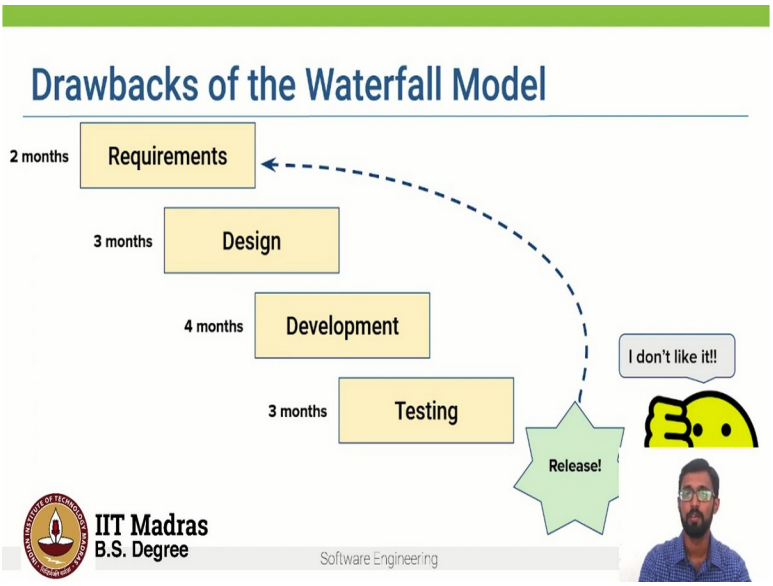
It is known as waterfall model, because each of these phases occur one after the other. And it is known as a Plan and Document perspective because before we start writing code, we have to come up with a plan for the project, which includes extensive detailed documentation of all the phases of that plan. And progress is then measured against this plan.

(Refer Slide Time: 4:02)



So, learners, let us reflect on this question. So, we saw that in the waterfall model, each of these phases follow one after the other. So, what could go wrong if we follow these phases sequentially? Please pause this video and think about some reasons before proceeding.

(Refer Slide Time: 4:35)

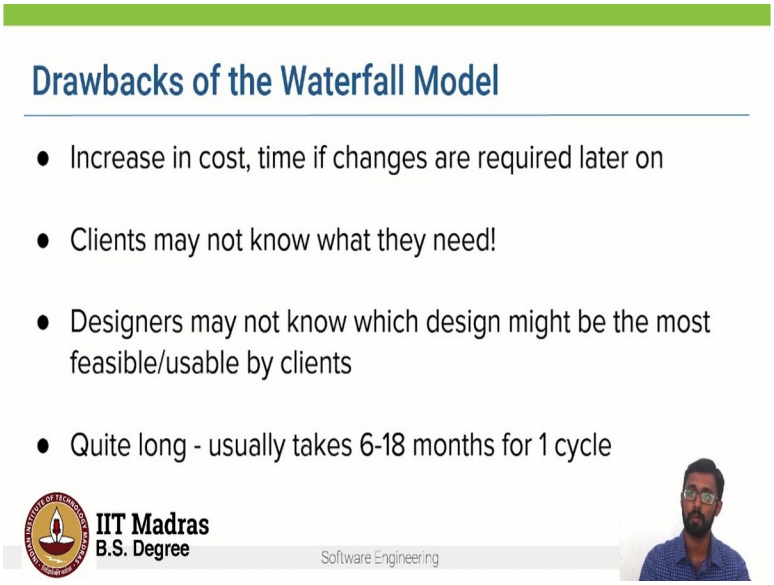


So, now let us look at some of the drawbacks of the waterfall model. Let us say, we are developing a software for a client and we talk to the client we gather requirements. Let us say that takes around 2 months. Then based on these requirements, we come up with the big picture view the design that takes around 3 months.

Then after that we start writing code for the design that takes around four months, and we test and fix bugs, which takes another three months. So, from the time of gathering requirements till the testing phase and finally the release, it takes around 12 months, it takes 1 year to develop the entire system and we release it to the client. But unfortunately, the client says that he or she does not like it.

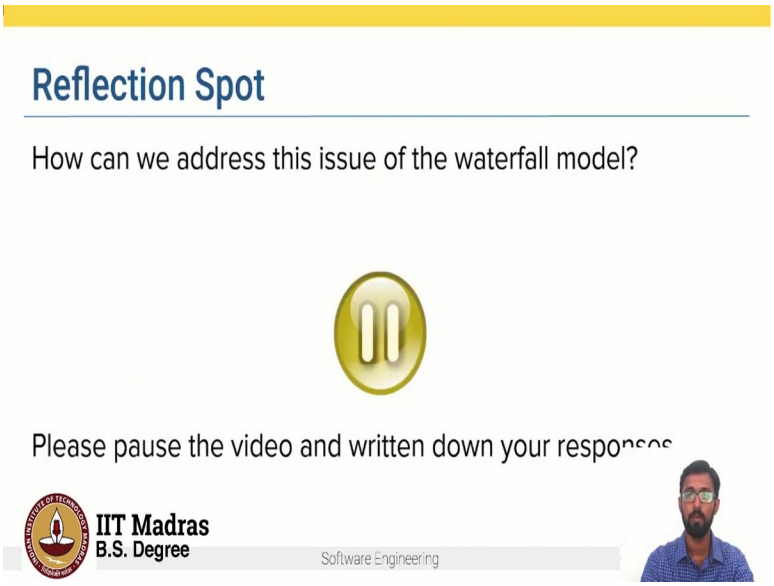
There has been some misunderstanding regarding what the requirements are certain regarding certain aspects of the requirements or maybe the client changed his mind about certain aspects of some requirements, maybe they do not like the user interface. So, these are some drawbacks, and because of this, now, I have to go back and refine my requirements and start this process again. So, what are the drawbacks of the waterfall model?

(Refer Slide Time: 6:03)



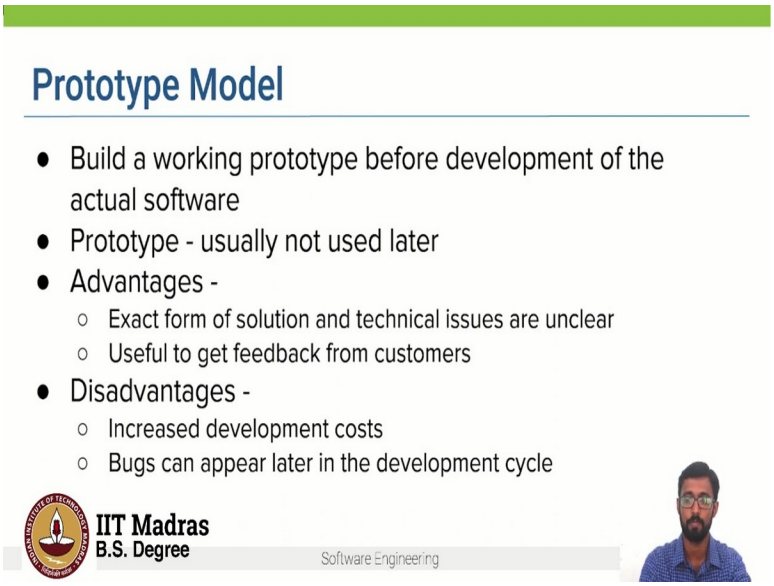
One as we saw, so, there is an increase in cost and time, if changes are required later on, as we saw in this scenario. Second, clients themselves may not know what they need. So, they might need to see something before they even realize what they actually need. And third, even the developers or the designers, they might not know which design might be the most feasible or most usable by the clients. And finally, as we saw one pace or one iteration can take maybe 6 to even 18 months for one cycle to occur. So, these are certain drawbacks of the waterfall model.

(Refer Slide Time: 6:58)



So, now, as we have seen, there are certain drawbacks of the waterfall model, how can we address these issues or these disadvantages of the waterfall model? Can you think of some solutions? Please pause this video and think about some solutions before proceeding.

(Refer Slide Time: 7:28)

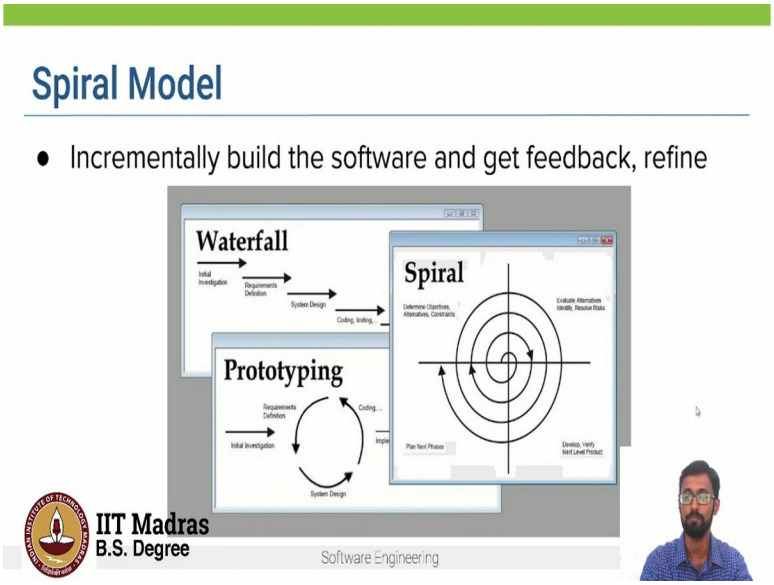


So, one solution can we get feedback from the customers earlier on. So, in this model, which is known as the prototype model, we build a working prototype of the system before development of the actual software. And this prototype has limited functional capabilities, and the performance might not be optimal compared to the actual software. The advantage is that it is easier for the customers to understand what they want, once they see this prototype as well as for developers and engineers to understand how to build it better once they have done it for the first time.

And usually the prototype it is not used later it is thrown away or new software is built. So, what are the advantages of the prototype model? So, one, it is useful when the exact form of the solution is not known and even the technical issues are unclear it is good to build a prototype and as we saw it is useful to get feedback from customers.

However, what is the disadvantage? One is that it can increase the cost of development for projects because as we saw the prototypes are not usually used later on and since we are not building the actual system, so what about risks and bugs which appear later in the software development in the development cycle. So, the prototype model cannot capture this.

(Refer Slide Time: 9:22)



Another solution can be to incrementally build the software, get feedback, refine and this module is known as the spiral model. So, the spiral model it combines both the waterfall and the prototyping module. And as you can see in this image, there are four phases. First, we determine the objectives, alternatives, constraints for this one iteration. And then we evaluate alternatives we resolved risks and then we develop and test and release the product to the client, and then we plan for the next phase. So, at each iteration, there was a refinement of the prototype compared to the previous version.

And unlike the waterfall model, where we document everything at the beginning, the requirement documents are developed across each iteration as and when they are needed. And the software eventually evolves into the final product. And we also involve customers at the end of every iteration and this helps in reducing misunderstandings between the client and the developers. But one drawback of this model is that the iterations still take long time so it takes around 6 to 24 months for each iteration.

(Refer Slide Time: 11:07)

So to summarize, in this video, we looked at the software development lifecycle, which is the different activities that need to be carried out for software development. We looked at different models in this plan and document perspective. We looked at the waterfall model where each of the phases are followed sequentially.

Then we looked at the prototype model where we build a prototype first and get feedback from users and the spiral model which combines the advantages of waterfall and the prototype model. Now, in the next video, we will be looking at the Agile perspective, which is a quite popular perspective nowadays to develop software. And we will be using many of the principles from the Agile perspective throughout the course.