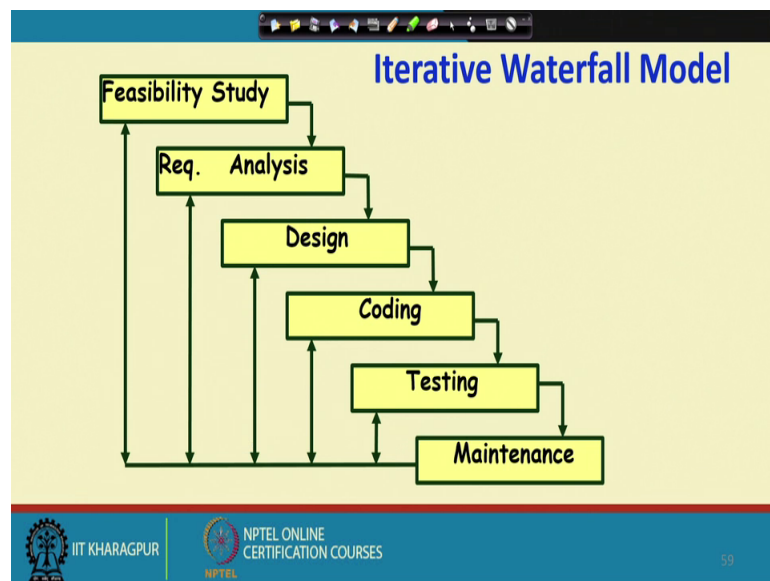


Software Engineering
Prof. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 09
Waterfall Derivatives

Welcome to this lecture, in the last few lectures we have been looking at the life cycle models. We looked at the classical waterfall model which is the basis for all other models, but then the classical waterfall model is hard to use in a project. The main problem is that it is idealistic and it has no way to accommodate corrections to the work products. It is just a pure waterfall model. But 100s and 1000s of mistakes occur during the development and the iterative model; iterative waterfall model overcomes this issue with the classical model and provides feedback paths.

(Refer Slide Time: 01:28)

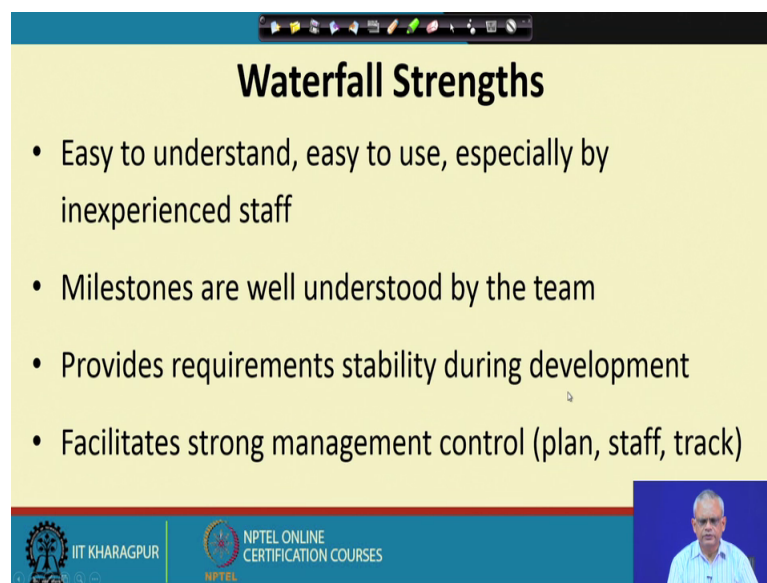


We were discussing about the iterative waterfall model in the last lecture; now let us get started at from that point. The iterative waterfall model it provides feedback paths and a mistake in any phase; if it is detected later on, there is a way to correct those mistakes and also redo the subsequent phases. The iterative waterfall model was hugely popular in 1970s and 80s; it was used in almost every development work. But then it had some difficulties for which the newer life cycle models came up; actually the difficulties were

not failed those days because the projects were like that for which waterfall model is ideal.

But slowly the characteristics of the projects themselves changed as we were saying that the projects became sought and also service projects and so on. First let us look at what are the strong points of the waterfall model, why it was so popular? And then we will see the deficiencies of the waterfall model so that the newer lifecycle models would be easier to appreciate.

(Refer Slide Time: 02:56)



Waterfall Strengths

- Easy to understand, easy to use, especially by inexperienced staff
- Milestones are well understood by the team
- Provides requirements stability during development
- Facilitates strong management control (plan, staff, track)

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

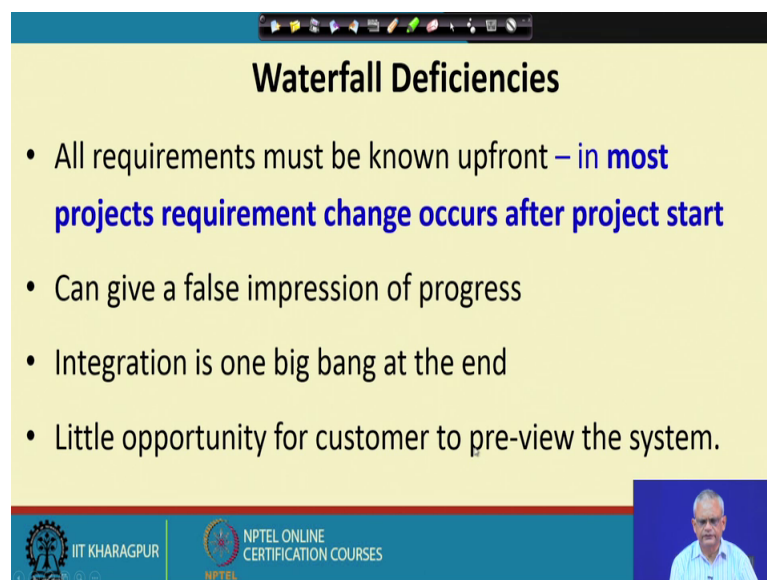
The strengths of the waterfall model include easy conceptually easy to understand and use. We had seen that it matches with our conceptual understanding of how software is developed. Even if the development staffs are inexperienced, they can easily understand the lifecycle model and start developing. In the waterfall model the milestones are well understood by every member of the team. We had seen that the milestones are basically the phase entry and exit; it also provides requirement stability during development. The requirement phase; the requirements are gathered and documented and after that there is no change to the requirements.

So, the developers are not interrupted they start to develop uninterrupted, they know the exact requirements and they start developing the software. But just imagine what would happen if they start developing and once in a while there is a change and they would

have to redo their work. And even the entire plan that they have made that would have to change; the overall design and so on.

So, the waterfall model provides requirement stability; once the requirement document is prepared the; it is not changed. And also for the project manager this is a very desirable model because the project manager can plan all the phases; how long each phase will take, when it will complete. And then can staff have the manpower for every phase and also can track it whether the project is proceeding as per the plan or take corrective action to put it back as per plan. So, far looks good the waterfall model has lot of strengths, but then there are several deficiencies of this model.


(Refer Slide Time: 05:35)



Waterfall Deficiencies

- All requirements must be known upfront – **in most projects requirement change occurs after project start**
- Can give a false impression of progress
- Integration is one big bang at the end
- Little opportunity for customer to pre-view the system.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES



Let us look at the deficiencies of the waterfall model; possibly the most problematic deficiency is that the requirements must be known upfront. The customer has to give all the requirements before the project starts and that is usually not feasible because the software is not there and the client has to imagine what all required. And it is very easy to miss requirements, very easy to give ambiguous requirements, wrong requirements.

And normally only when the user sees the software says that no; this is not what I wanted I wanted something else. In real projects, there is a large number of requirement changes required; the customer at the beginning of the project cannot visualize the exact requirements. And as the project proceeds the customer can say that see this is not what I

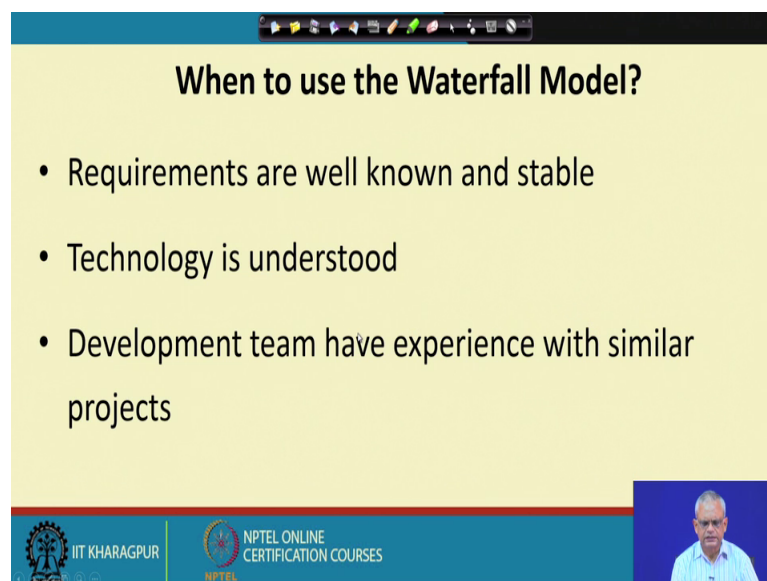
want and even the business or the user might change his methodology and. So, on he might want changes on that account also.

The second problem with the waterfall model is that gives a false impression of progress; that is the project manager may see that the phases are getting complete, the documents are getting ready and the project manager thinks that the project is progressing fine. But then the problem starts with the integration testing once the integration starts; then the problems appear that all the modules they need changes because the modules were developed thinking that the other functionality is their parameters and so on.

But then in reality it is not like that, there will be problem in integration. This is one of the major problem area here in waterfall model and the delay schedule delays starts from the integration and the project gets delayed further and so on. Another problem is that the customer is kept out of the development; the customer defines the problem and then just waits for the software to be developed. And once he gets the software, then he uses it and says no this is not what I wanted; he needs lot of changes to the software.

So, one of the problem in the waterfall model is that once the software is complete; it rarely meets the customer requirement and the reason is that the customer is kept out of the development work.

(Refer Slide Time: 09:01)



When to use the Waterfall Model?

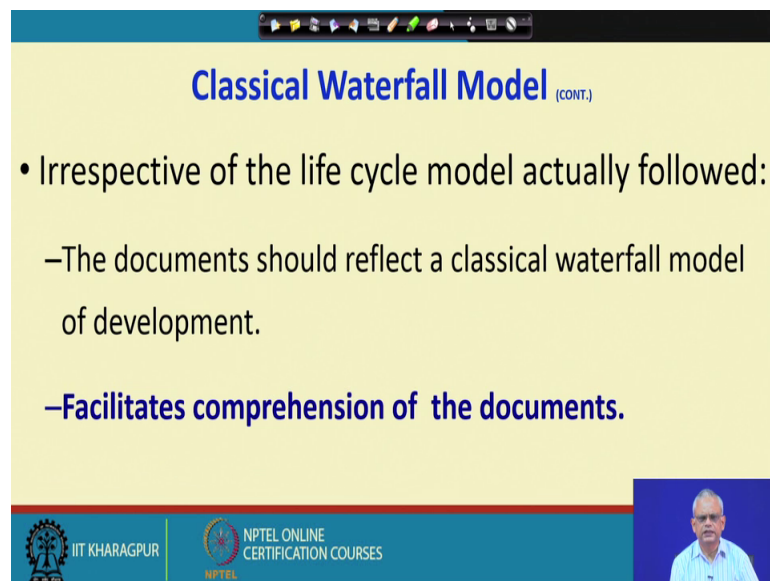
- Requirements are well known and stable
- Technology is understood
- Development team have experience with similar projects

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Let us see the projects for which the waterfall model is suitable; if the requirements are well known and stable; if many, software has been developed and this is just one of that. So, that it is well understood that what is required of the software and can be frozen upfront.

The technology is well understood; how the development will occur, what technologies will be used and so on. And also the development team is familiar with the software to be developed. If you look at these characteristics these are basically some software which exists and we just want to have a small change version of that. And the developers have already developed similar software, but then if we want to develop customized software; let us say we had a software and we just want to add a few changes to that, then the waterfall model will not be suitable.

(Refer Slide Time: 10:29)



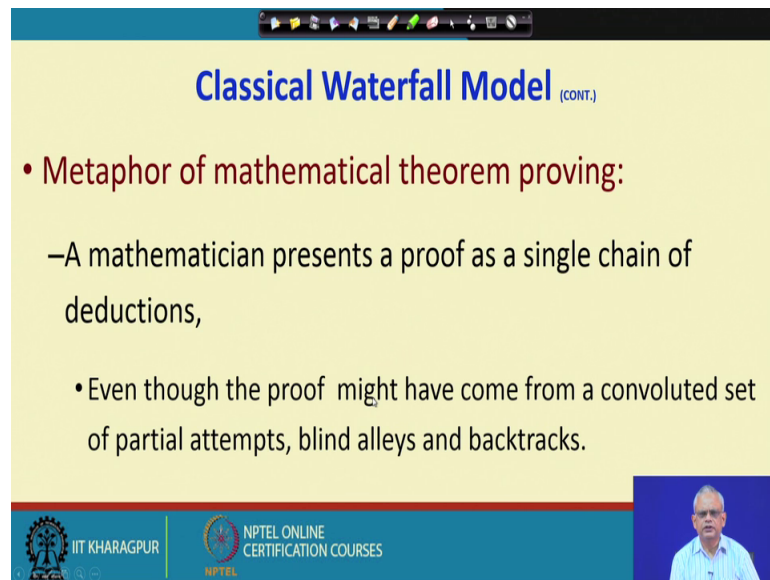
Classical Waterfall Model (CONT.)

- Irrespective of the life cycle model actually followed:
 - The documents should reflect a classical waterfall model of development.
 - Facilitates comprehension of the documents.**

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

What about the classical waterfall model? We said that it is not useful for real projects, but does it have any use at all? Ok one thing is that all documentation of a project is actually done as per the classical waterfall model. The development might have occurred using the iterative model or any other model, but then the documents are written as if the classical waterfall model was used. Let us understand why that is the reason.

(Refer Slide Time: 11:06)



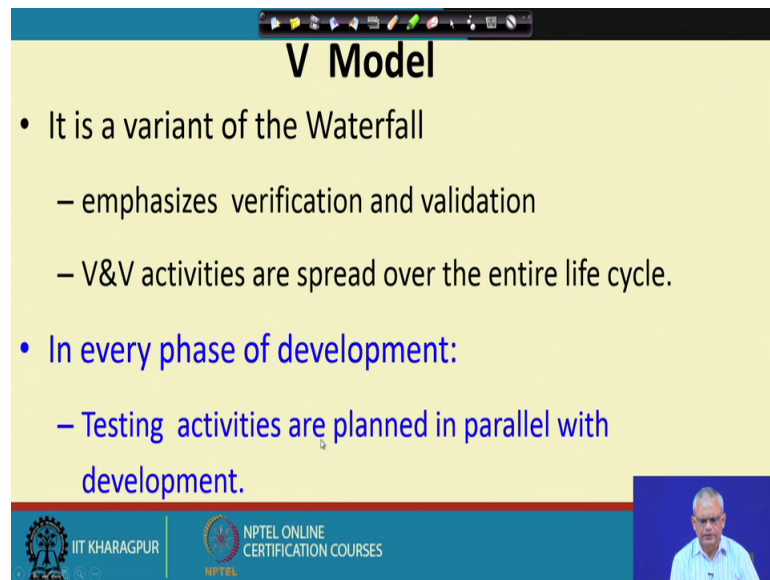
The slide is titled "Classical Waterfall Model (CONT.)" in blue text. It features a list of points in red and black text. At the bottom, there are logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, along with a small video inset of a speaker.

- **Metaphor of mathematical theorem proving:**
 - A mathematician presents a proof as a single chain of deductions,
 - Even though the proof might have come from a convoluted set of partial attempts, blind alleys and backtracks.

Just think of the way the mathematician proves theorem, given a problem that he wants to prove he would work in many directions, he will backtrack; he will change in between cross it out start again and so on.

But then when he finally, writes his theorem it appears as if a single chain of thought, all the mistakes that he had done the different alternatives he has tried etcetera, etcetera are not shown. Because that helps somebody to understand it well; if he had included all the mistakes that he did, all the wrong directions we took backtracked and so on; it would appear extremely confusing for somebody. And that is the precisely same reason why the documentation in software projects are written as if classical waterfall model is used because that way the documents can be easily understood by somebody trying to understand the documents.

(Refer Slide Time: 12:40)



V Model

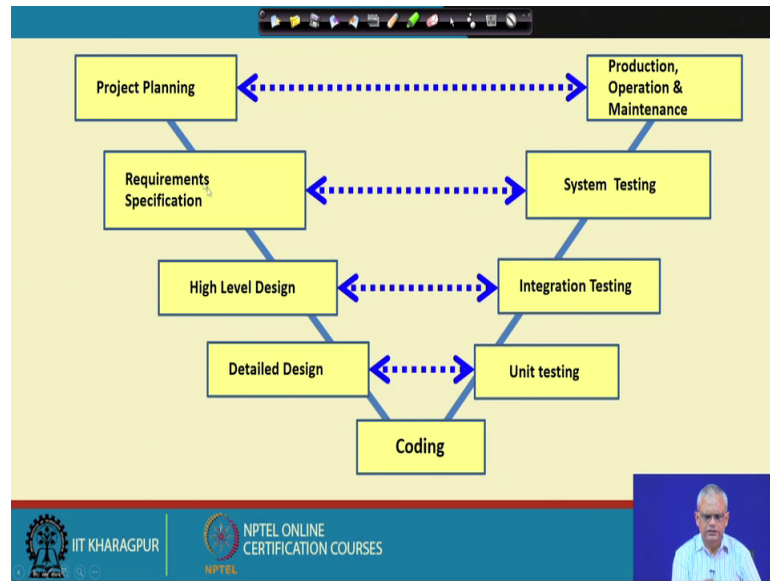
- It is a variant of the Waterfall
 - emphasizes verification and validation
 - V&V activities are spread over the entire life cycle.
- In every phase of development:
 - Testing activities are planned in parallel with development.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, let us look at a derivative of the waterfall model name is V model; it is a variant of the waterfall model and it emphasizes verification and validation. So, that really means is that it will be useful for software that are used for safety applications, where reliability safety are important; the V model is the one that is preferred. The verification and validation activities are spread throughout the entire lifecycle. If you remember in the iterative waterfall model it was only during the coding phase that unit testing is done and in the testing phase the integration and system testing is done.

But you are starting with the requirements and design all phases there are verification and validation activities. And the testing activities are planned in parallel with development as the different aspects of the project progressed; more accurate test case designs are done.

(Refer Slide Time: 13:59)



A visual representation of the model looks like a V shape and that is the reason why this model is called as a V model. Look at here that starting with project planning, requirement specification, high level design and then detailed design finally coding and then unit testing, integration testing, system testing and then maintenance.

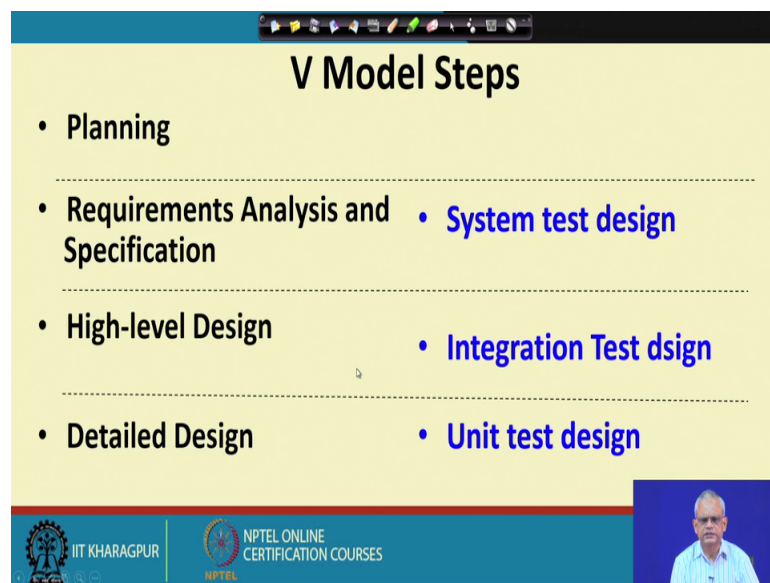
But in every phase; so, this part is the development phase on the left side and on the right side is the testing and maintenance. And if we look at the development phases during the requirements specification the plan for system test is done. So, that is the system test cases are designed during the requirements specification. Because after all the final software has to conform to the requirements; as far as the final software is concerned; the SRS document is taken, test cases are designed and then it is executed and checked whether it meets the requirements.

In the V model it is advocated that during the requirement specification phase itself; the system test cases should be written. There are two advantages to that; one is that testability of the requirements is kept in mind; how exactly it will be tested? During the development we are clear that how it will be tested? And another issue is that in the iterative waterfall model, what do the testers do during the development phases, do they join only during testing phase? No; not really they are part of the team.

So, what do they do? During the testing requirement specification design, coding etcetera; what do the testers do? The V model provides a solution to that and the testers

are actually busy here during the requirements specification, they are busy writing the system test cases. During the high level design, they are busy writing the test cases for integration testing and planning for integration testing; during the detailed design they are busy developing the unit test cases. So, during every development phase; testing is kept in mind and test cases are already developed. So, it becomes easier for the implementers that is for coding and so on is to know that; what is expected and they become conscious of the work that do that it has to finally satisfy those test cases.

(Refer Slide Time: 17:19)



The slide titled "V Model Steps" lists the following development phases and their corresponding test activities:

- Planning
- Requirements Analysis and Specification
- High-level Design
- Detailed Design

Corresponding test activities are listed on the right side of the slide:

- System test design
- Integration Test design
- Unit test design



The slide also features logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES at the bottom, along with a small video inset of a speaker.

If we look at the different steps of the V model, we see that during the requirements analysis and specification some test activity is done; that is system test case design and system test planning. during high level design the integration test case design and integration planning is done, during detail design the unit test design is done.

(Refer Slide Time: 17:58)

V Model: Strengths

- Starting from early stages of software development:
 - **Emphasizes planning for verification and validation of the software**
- Each deliverable is made testable
- Easy to use

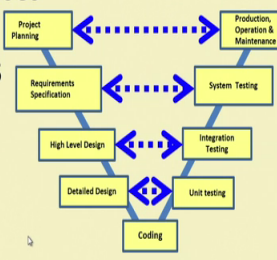


The strength of the V model is obvious that it emphasizes verification and validation. Every deliverable is made testable for example, during requirements specification testing is made kept in mind and therefore, the requirements become more testable. And also the model is easy to use; it is after all a derivative of the waterfall model, very similar to waterfall model accepting that the V and V activities the verification validation are spread throughout the lifecycle.



(Refer Slide Time: 18:38)

V Model Weaknesses

- Does not support overlapping of phases
- Does not handle iterations or phases
- Does not easily accommodate later changes to requirements
- Does not provide support for effective risk handling



The diagram illustrates the V Model Weaknesses. It shows a V-shaped structure of development phases on the left and testing phases on the right. The development phases are Project Planning, Requirements Specification, High Level Design, Detailed Design, and Coding. The testing phases are System Testing, Integration Testing, and Unit testing. Dashed blue arrows indicate the flow from development to testing. A double-headed dashed blue arrow at the top connects Project Planning and Production, Operation & Maintenance, suggesting a lack of support for overlapping phases. A double-headed dashed blue arrow between Requirements Specification and System Testing suggests a lack of support for iterations or phases. A double-headed dashed blue arrow between High Level Design and Integration Testing suggests a lack of support for later changes to requirements. A double-headed dashed blue arrow between Detailed Design and Unit testing suggests a lack of support for effective risk handling.

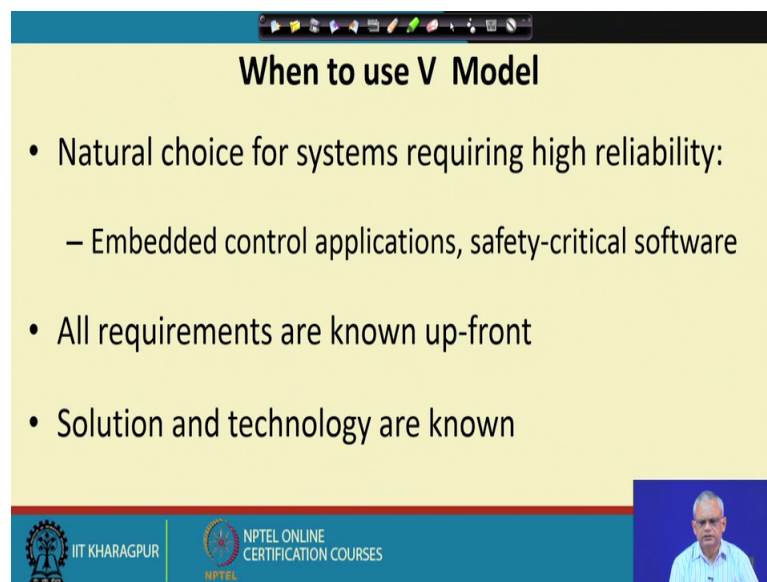


Now, what about the weaknesses of the V model? First is that it suffers from the same weakness as the iterative waterfall model; that is the requirements are frozen before the project starts. And there is no scope of changing the requirements later on it is just a waterfall model. Another issue with the waterfall model and also the V model is that; it does not support overlapping of phases. This is the problem even for the waterfall model even though we did not mention it.

The problem is something like this that let us say there are 10 designers working to design. They have designing different parts of the software; now let us say one designer completed early and some designers take much more time. So, the one who completed early what does he do? Does he just wait for the phase to complete? So, that he can start the next phase.

If the model is followed as it is then there is a lot of idle time because it does not support overlapping of phases. Iteration is not shown here, in the V model iterations are not iteration between phases is not shown, does not accommodate requirements change, risk handling is not provided.

(Refer Slide Time: 20:34)



The slide is titled "When to use V Model" and is presented in a yellow box with a blue header and footer. The header contains a navigation bar with various icons. The main content area lists three bullet points: "Natural choice for systems requiring high reliability:" followed by a sub-bullet "Embedded control applications, safety-critical software"; "All requirements are known up-front"; and "Solution and technology are known". The footer contains the logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, along with a small video inset of a man in a blue shirt.

When to use V Model

- Natural choice for systems requiring high reliability:
 - Embedded control applications, safety-critical software
- All requirements are known up-front
- Solution and technology are known

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

It must be clear that the V model is used when there is a high emphasis on safety reliability embedded such as the embedded control applications.

And the requirements should be known upfront and the solution and technology should be known. And if these are the characteristics of the project then V model is a good model. Now let us look at another variant of the waterfall model which is called as the prototyping model.

(Refer Slide Time: 21:13)

Prototyping Model

- A derivative of waterfall model.
- Before starting actual development,
 - A working prototype of the system should first be built.
- A prototype is a toy implementation of a system:
 - Limited functional capabilities,
 - Low reliability,
 - Inefficient performance.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

The prototyping model, if you as you can see it is very similar to the waterfall model the small difference here is that to start with there is a prototype construction and the requirements phase is not there actually; it is there is a prototype construction.

It is a small change of the waterfall model before the development design coding etcetera starts a prototype needs to be built. But what exactly is a prototype? A prototype is a toy implementation of the system, but then how do we get a toy implementation of a system? Let us imagine that we are trying to implement a function, for a toy implement of the function we might have a table stored inside the function; that just looks up the values from that is required for the function, for which input value what will be the output.

So, for restricted input values it will work; we have stored that in a table; it would have a limited functional capabilities, does not work for many inputs, functions are simplified as much as possible. For example, by using a tables to store the values that are expected, low reliability, inefficient performance; but then why do we need to develop a prototype? Need to develop a prototype because we can show it to the customer.

The customer to start with it is very hard for the customer to visualize the entire software and that is one of the reason why the waterfall model is unsuccessful in many projects. It finally, does not match the customer's requirements, but since in the prototyping model before starting the development. We are creating a prototype of the software. The customer will get a better feeling of how the software; finally, will appear and he might suggest changes and that is incorporated into the prototype and until the customer says that is exactly meets his requirement, the prototype refinement continues.

Another reason why a prototype needs to be constructed is that sometimes the developers do not know that whether some technical issues can be met. For example, let us say the transaction rate required by the customer is very high and the developers do not know whether the database will actually be able to provide that sort of transaction rate and whether it can be updated on the browser.

By constructing a prototype the developers can actually test out that whether they are able to meet the required transaction rate. So, for developing the prototype there are two advantages; one is that the customer gets a feel of the system and can modify any requirements. The second is technical; the developers might want to experiment with some aspect on the technicalities and then they can decide which way to start the construction.


(Refer Slide Time: 25:46)

Reasons for prototyping

- **Learning by doing:** useful where requirements are only partially known
- **Improved communication**
- **Improved user involvement**
- **Reduced need for documentation**
- **Reduced maintenance costs**

The diagram illustrates a waterfall model with stages: Prototype Construction, Design, Coding, Testing, and Maintenance. Arrows show a downward flow from one stage to the next, with feedback loops from Testing back to Design and from Maintenance back to Design.

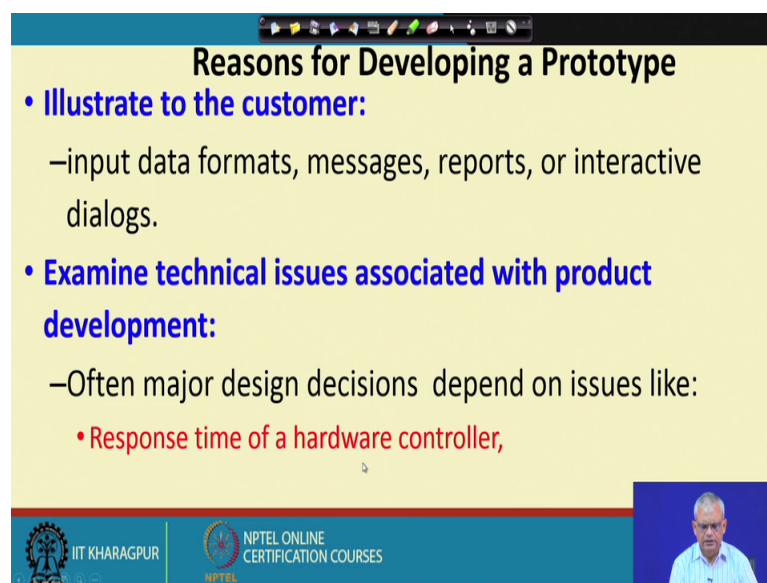
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES



There is another reason why prototyping is useful is that the developers learn by doing the work. So, the second once the prototype is complete they start the development and they can do a better job. And the user also by looking at the prototype can finalize the requirements. That is improved communication with the customer in the waterfall model, pure waterfall model the customer feels isolated for the entire development; the customer has to just wait, but here could have see the prototype and how the system will look like finally.

It also reduces the need for documentation and also typically a prototyping model. The maintenance costs are low because the quality of the development is good here because once the prototype is constructed; the developers have experience with the software and they can do the actual development in a much better way and that is the reason why it will incur less maintenance costs.

(Refer Slide Time: 27:20)



Reasons for Developing a Prototype

- **Illustrate to the customer:**
 - input data formats, messages, reports, or interactive dialogs.
- **Examine technical issues associated with product development:**
 - Often major design decisions depend on issues like:
 - Response time of a hardware controller,

The slide features a yellow background with a blue header and footer. The footer contains the logos for IIT Kharagpur and NPTEL Online Certification Courses. A small video inset of a speaker is visible in the bottom right corner.

The customer is illustrated about the software can make up the mind can specially examine the user interface issues the format of the display interactive dialogues and so on. And the developers themselves can examine technical issues for example, response time of a hardware controller. So, these two; I would say that these are the major advantages of prototype model that the customer can get a view of the software and also the developers can examine technical issues.

We will stop at this point and we will continue in the next lecture.

Thank you.