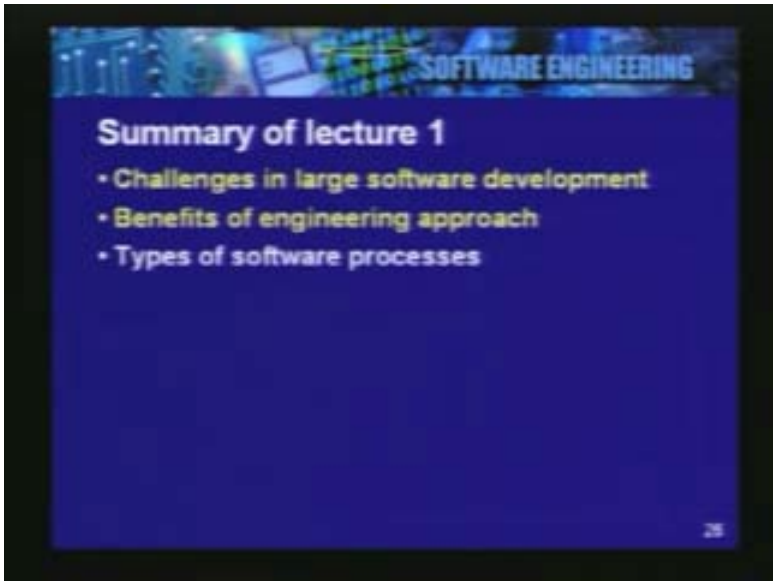
Software Engineering
Prof. N. L. Sarda
Computer Science & Engineering
Indian Institute of Technology, Bombay
Lecture - 2
Introduction to Software Engineering
Challenges, Process Models etc
(Part 2)

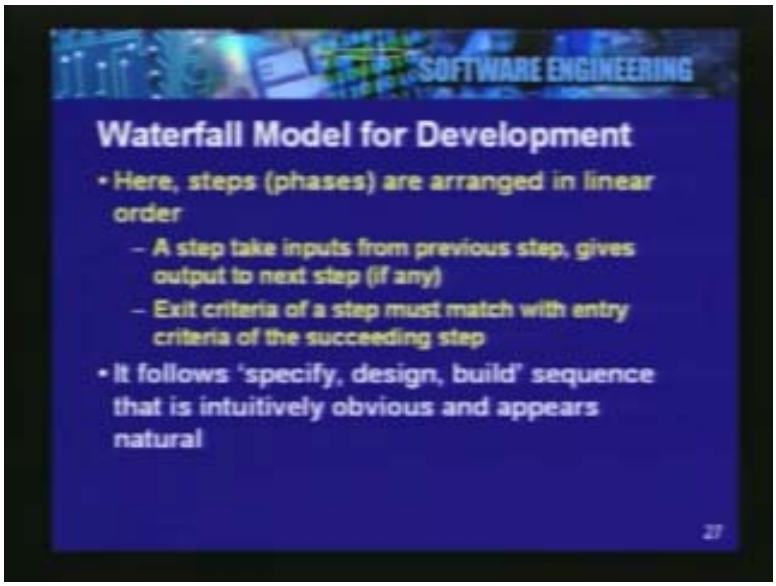
This is second lecture on introduction to software engineering. Let us start by summarizing by what we saw in the first lecture. We understood the challenges in developing the large software. We want to do it in given time, within given resources and meet user's requirement. This is indeed very challenging. We then saw the benefits of engineering approach which consists of defining different milestones, different stages in the development and carrying out periodic review and having a well-defined methodology for the development. Then we have talked about different types of software processes. These are processes for development of software, for project planning and management, for configuration management and then for managing the software processes themselves, because they are not fixed and static. They keep changing. So any large software development organization will need to invest time and resources in keeping the software processes up to date and state of art.

(Refer Slide Time: 01:52)



Then we defined a typical step in a software development process we identified that the step will consist of some entry criteria, exit criteria, it will have some specific task to do, it will take certain inputs, it will provide some outputs and also the step will be under management control. So this is a very important definition of the typical step in the software development process. Then we also saw characteristics of good processes. And one of the most important characteristic was that it should be predictable, it should be well defined, and it should be repeatable. This is primarily what we focused on in the first lecture. And towards the end we began to look at the waterfall model for software development. We briefly introduced this last time. We said that a waterfall model consists of many steps organized in a linear order. So that one step produces the right kind of outputs for the next step.

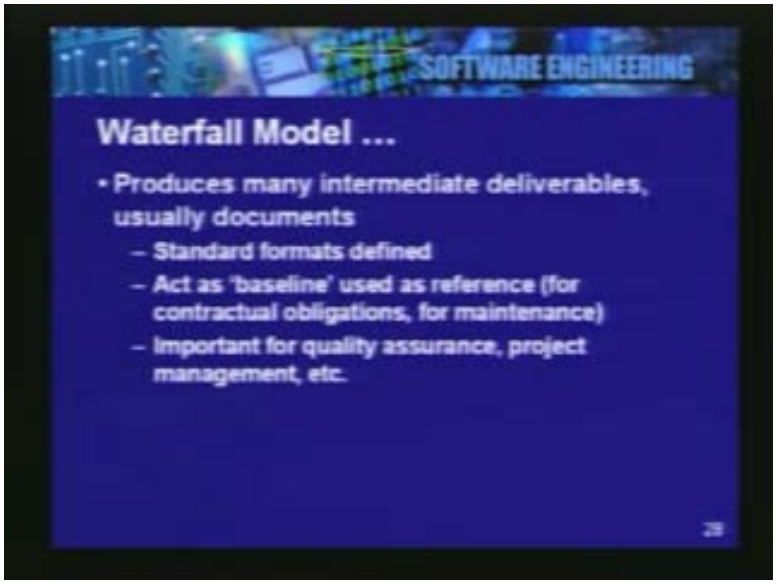
(Refer Slide Time: 03:34)



There is a linear or sequential order in which the different activities are performed. It follows what we may call; "specify, design, build" sequence of activities, which should appear to be quite obvious. What it really means is that, you cannot design something which is not clearly specified. And if you have not done the design, naturally you cannot build. This is like going to market without knowing what you want to buy. Naturally you can't buy them or you can't compare different products. You need to have clear specifications in mind before you go and want to buy something. Similarly when you want to build something, you must define what needs to be build, what are the functions that need to be performed, and then carry out a design which is efficient, effective. And finally only when the design or the blue print is done, we start building. So we say that this sequence of activities appear to be obvious and natural. This is what the waterfall model tries to do.

Waterfall model produces many intermediate deliverables. There is a deliverable at every step in the model. For each of these, proper standards have been defined. Most of these deliverables are actually documents which have a standard format.

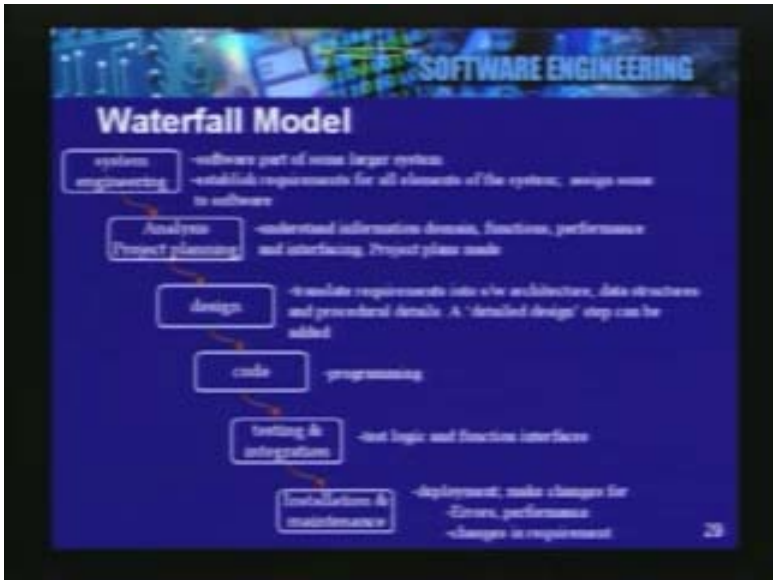
(Refer Slide Time: 05:00)



Many of these documents will act as a baseline for future reference. In fact some of these baselines are in the nature of contractual obligations. You may be an independent agency who is doing software development for a client. So there has to be a baseline, there has to be a reference which clearly defines what your obligations to the customer are and what kind of product you should make for him. So such kind of baselines would be produced in the process of development. And waterfall model defines such baselines or other deliverables or other documents which are required for subsequent steps. One important point about this deliverables is that, once a step is completed, the output of that step is clearly defined in the form of a document, which can be then handed over to people who will perform the next step. So this model facilitates change of people from step to step. We are not making it necessary that the same people who have done the analysis should do the design; the same people who have done the design should do the implementation. In fact we saw that in the example of civil engineering, we have different people with different expertise and different people are called upon to do certain specific things at different points in the life cycle of a project. Same thing happens in a software development. We define these documents, they are of certain formats, and they are handed over to the next team who will perform the next step. So there is an interchangeability of the people and different steps can employ people of different skills as necessary for that step.

These documents also play an important role for quality assurance, project management and so on. This methodology is also very widely used. There are some variations possible, but it is commonly followed because of the natural development cycle on which it is based. Especially when the requirements can be clearly defined, waterfall model is an ideal model for development software. So let us see what the different steps are.

(Refer Slide Time: 07:35)



This diagram identifies different stages in the waterfall model. Let us look at each one of them briefly. Then later on we will look at each of them in some more details. So the first step is the system engineering step where you try to understand the overall problem and identify which of the problem elements are to be handled through the software. It is possible that certain requirements of the problem may need to be done manually, because automation may not be possible whereas some other functions will be performed by the software. So one of the goals in system engineering is to understand the overall context of the problem. And then identify what would be the responsibilities to be met through the software. After doing that we undertake the analysis phase. In the analysis phase we understand the problem domain of the user in more detail. We understand what kind of information is involved what kind of data is involved, what functions need to be performed, what are the performance and interfacing requirements. This part is the analysis part whose purpose is to identify or clearly define the requirements which are to be met thru the development of the software along with.

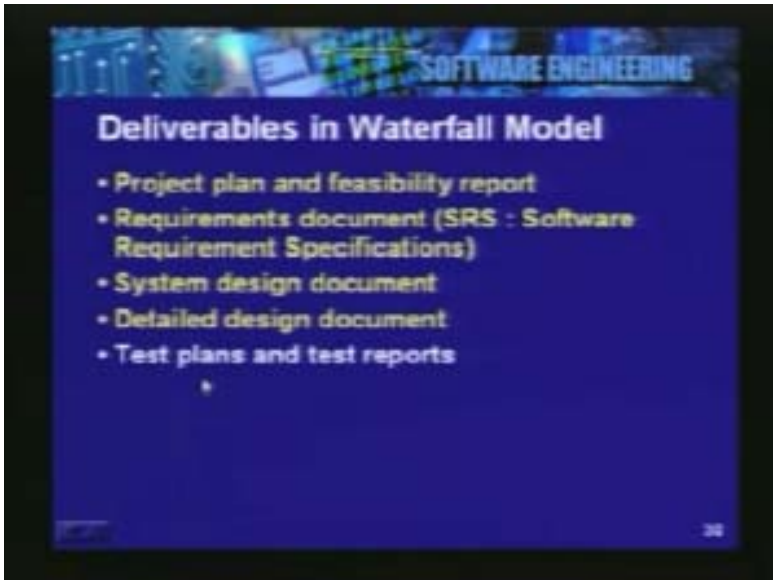
In the analysis we also try to carry out project planning either as a part of the same step or if it is large project, the project planning step may be made as a separate step. And the purpose of project planning is to identify how we will carry out the project what are the different steps, what are the deliverables, and what would be the time frame what could be the resources allocated. That is what a project planning is all about. After the analysis is done, the next step leads to the design step. In the design step we translate the requirements into the software architecture when we prepare the database design and so on. So this is a primarily a technical step and in this step the requirements are translated into an implementation framework, which will be implemented in the next phase that is the coding phase, in which the actual programming and implementation will be done.

So you have a design and the design is followed by implementation. In fact as it notes here the design itself may be divided into two stages called high level design and detailed design. Again this would depend on the complexity of the task. After you have done the implementation or carried out the programming phase, when the individual units of the software are ready, you would get into the next phase called testing phase and integration phase. In this all the individual pieces of the software are individually first tested, and then the whole software is put together. The objective is to test first the logic of individual modules. And then when we put all of them together we want to test their interfaces, so that they work collectively in a manner in which they were intended to work. This is the testing and integration phase and this is followed by actual deployment of the software. Now the software is ready. It is deployed and implemented. The user will take control and will start using the software.

After the deployment there is an ongoing maintenance phase. In the maintenance phase, we have to make the necessary changes; if any errors are encountered we will have to remove the errors. If the software is not meeting the performance requirements we may have to review the software, maybe we have to go back and look at the design whether the design can be modified or if there is something else that can be done in order to improve the performance. So this is an ongoing step which may generally be called a maintenance phase. And in this phase, the software is either enhanced or is perfected or is ported to new hardware or new machines. This is an ongoing activity which needs to be done throughout the life time of the software. These are the different phases through which a typical software development goes through. We have identified here the main steps. Each of the steps has certain deliverable. Let us enumerate the different deliverables which come out in a given sequence when we perform these various steps.

One of the important deliverable is the project plan. We will be carrying out the project according to the plan that we make. Of course the plan needs to be revised periodically as we progress through the project. One of the deliverable could also be the feasibility report. The feasibility report is a report which defines the cost effectiveness of the approach and the whole software project, whether it is a good investment we are making, whether the software will be cost effective, whether it will give the kind of benefits we are expecting from the software.

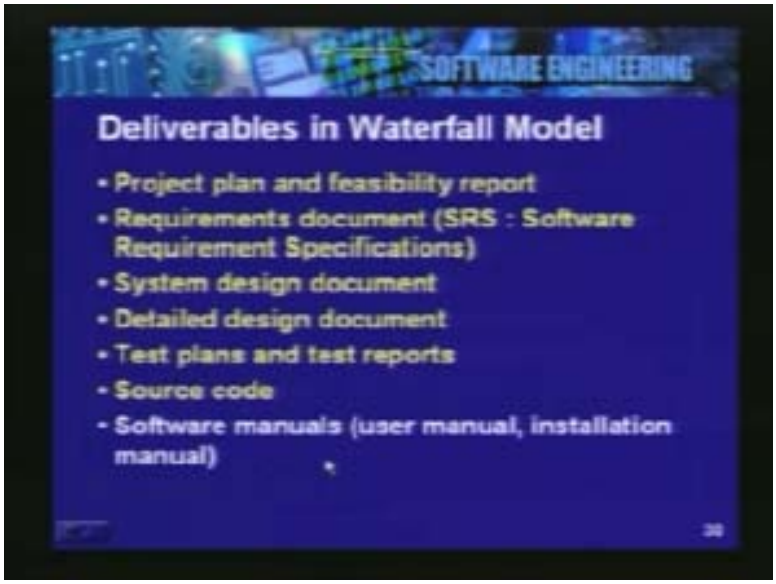
(Refer Slide Time: 14:48)



So the project plan is one deliverable, feasibility report is another deliverable. One of the important deliverable is requirements document which is also called software requirement specification or SRS for short. SRS is very important baseline and defines the functions that the software will perform for us. Obviously the SRS comes from the analysis step. The next step is the design document. It defines the different components of the software and how they will be implemented. You may have a detailed design document. This document may specify individual units which will be implemented by different programmers. We may have then specific test plans and expected test reports. What it implies is that testing is not a on the fly kind of the task. Tests are planned well in advance. They define what inputs will be given to the system and what outputs are expected and whether the expected outputs are correct and the performance is as expected. So test plans are made well in advance and as soon as the software is ready the test plans are carried out and the workability of the software is demonstrated through the test plans.

Of course the source code is an important deliverable and a deliverable which is in executable form. Unlike other deliverables which are documents, this is a deliverable which actually would be loaded on the computer and will be executed which actually defines the software contents. And then we will also have different types of manuals or documents which will be important for users and for administration of the software, for installation of the software and so on.

(Refer Slide Time: 15:49)



So these different types of deliverables are identified from different stages or different steps in the waterfall model. At the end of each step, if you recall we had said that every step ends in a validation and verification step. Similarly in the waterfall model when we move from one step to another there would be such reviews. Each step would end in a review. These review reports are very important inputs for identifying whether the development proceeded as per the plans, whatever the short comings whether the defects will were detected and were properly addressed.

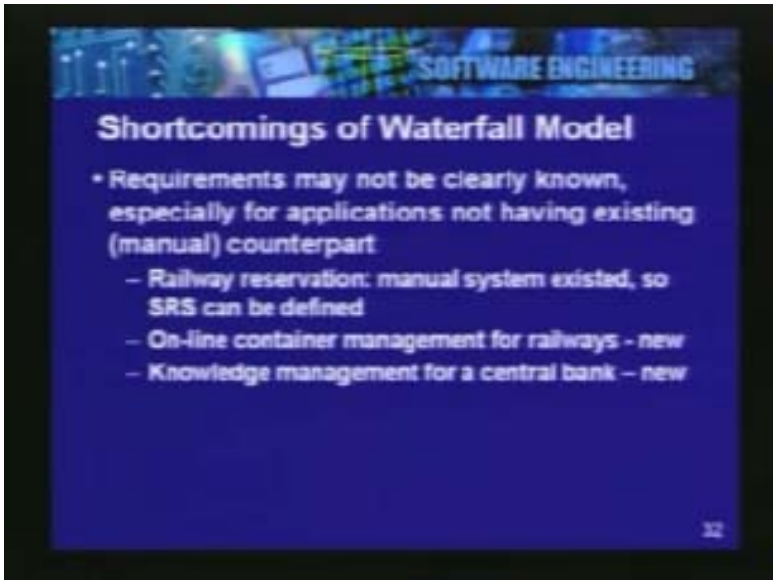
So review report is basically a catalog of how the development went through and it is an important input to the development team as well as to the management. So these are different deliverables. Many of them are important of which SRS is obviously one of the baseline. Design document is also very important for future maintenance. Source code definitely is important as it is executable and without this user manual, installation manual, some administration manuals and the system cannot be effectively used. So as you see here these different documents are essential part of the software. So software is not just the source code but all this documents which accompany software. Now as we go through this waterfall model and we perform the different steps, naturally we are investing more effort. The diagram shows the cost effort distribution as we run through the different steps in the waterfall model. So there are different steps are shown here.

(Refer Slide Time: 17:41)

As you see that some of the earlier steps have smaller effort than the subsequent steps. For example the detailed design implementation is very effort intensive steps. The line here is a cumulative task and as you see it increases rapidly as we progress in the development. Accumulated cost increases dramatically as we go from initial steps to later steps. Because as we move along the steps we are committing the time of programmers, technical people and computer time, all this is naturally a cost factor. This effort distribution is important for us to understand. Because it clearly brings out that we must take all efforts to remove errors as early as possible in the lifecycle. We must discover and fix these errors as early as possible.

Because fixing them subsequently will be a very costly operation. So suppose there is a misunderstanding or the analysis phase has not clearly identified the functional requirement, now that if it is not detected at the review phase of the analysis, it would be carried over design, detailed design and implementation. And it is possible that we discover that mistake during the operational part of the software; when the users have started using the software. In order to make the correction we will have to undo the code design etcetera and go back right up to the analysis phase and identify what was missed out what was wrongly understood, and carry out the design and implementation and testing process again. So it is important that such errors are captured as early as possible and they are not allowed to slip into the subsequent steps. So in any software development approach it is important to understand, how the cost and efforts are distributed, not only for project planning, but for emphasizing an error free activity in each step.

(Refer Slide Time: 20:55)



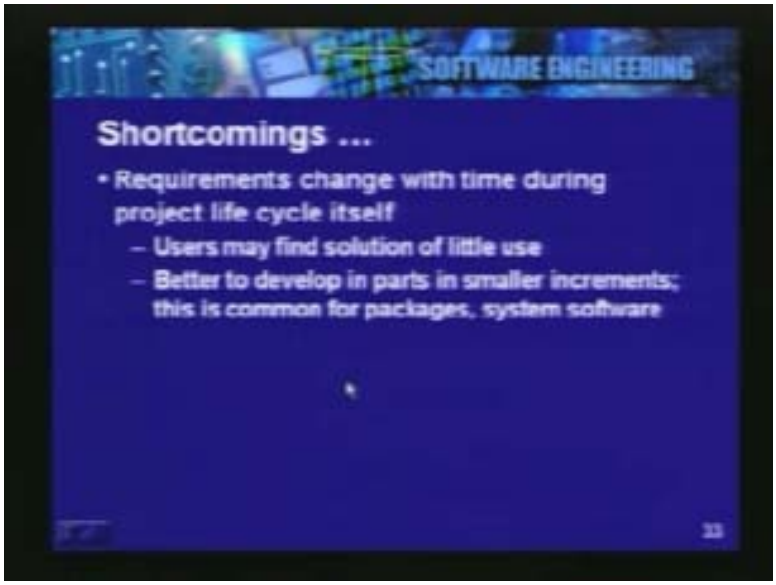
There are some shortcomings in the waterfall model. One of the main criticisms of waterfall model is that it goes linearly and it assumes that all the requirements can be clearly defined during the analysis phase. There are many situations where requirements may not be clearly known. This often happens when there is no manual counterpart for the problem that we are solving. If a manual counterpart exist, that means the function business function is already been carried out manually and we are planning to automate that, then in this case it is easy to define the requirements. But in a situation where there is no such manual counterpart, when the system is been planned first time, then the requirements may not be clearly known.

Here is an example. We know that railway reservation has now been automated fully. Before that, however the manual system existed. We knew what is the role or what are the tasks involved in the railway reservation. You want to reserve a seat on the particular train, you may want to cancel it, you may get waitlisted, waitlists have to be arranged and there is a method for arranging. All those things were already defined as functional activities. And therefore one can say that as far as the railway reservation is concerned the manual system existed, and therefore it is possible to clearly define the requirements in this case. But there can be other examples where there may be no manual counterpart. For example, say a central bank wants to build a knowledge management system.

It believes that there is so much knowledge that is available with its employees in order to carry out their day to day task. It would like to capture this knowledge, build a repository, so that this knowledge is easily available to all other people in the organization. Now this is something which is quite new and it is difficult to define a SRS for an application like this. There is obviously some kind of uncertainty involved in terms of what it will consists of.

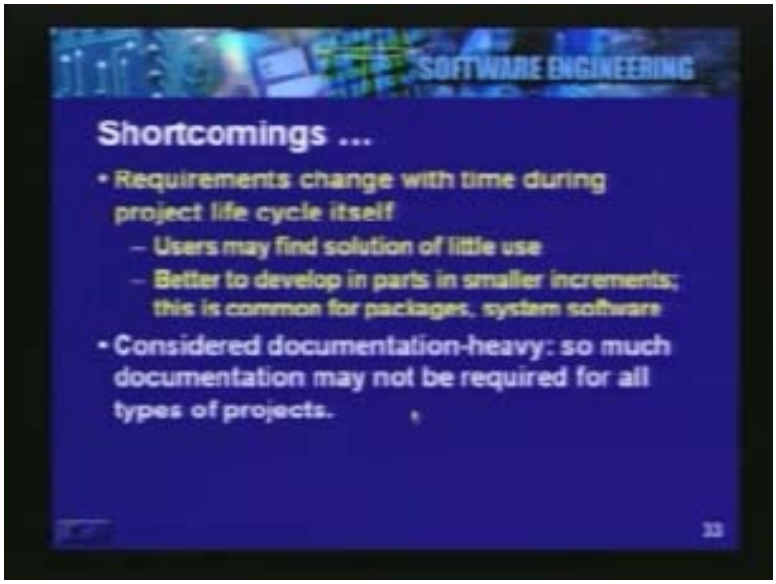
So this is clearly one issue when we try to apply for the waterfall model, Which the requirements may not be clearly known and it may not be possible to carry out analysis phase once for all and then never come back for revising the requirements. The other issue is that the waterfall model assumes that the requirements are quite stable and that they do not change with time. Now, this is not the case because the software development itself may be a long process, requiring months or years and during this time one cannot assume that the requirements will not change.

(Refer Slide Time: 24:00)



So in the process what may happen is that when the software is finally built, the users may find that it does not meet their requirements. In such a situation it is probably a good idea to develop the software in parts or in smaller increments and then release them and get the feedback of the user. So there may be different approaches rather than using a strictly linear waterfall model in such situations. Waterfall model is also considered very documentation-heavy, what it means is that too much documentation is being produced as part of the project. This may not be really necessary in all types of projects. So other models have been proposed in order to take care of some of the short comings or to handle some specific situations in project development scenarios.

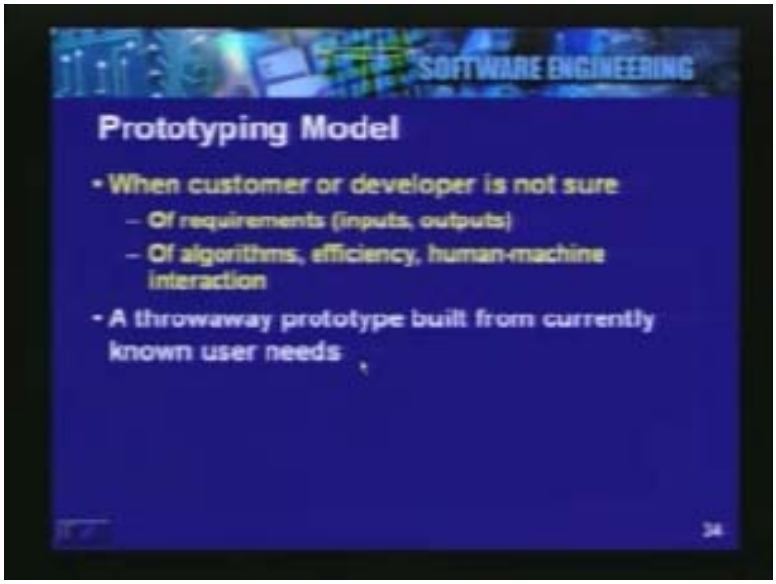
(Refer Slide Time: 24:54)



One of them is the prototyping model. In the prototyping model we try to build a prototype rather than the final solution. And we do this especially when the customer is either not aware of his requirements that what exactly he would like to how the software would do for him. Or the developer is not sure about how to develop the software, how to implement the business rules which are generally required for the software. So he may not be certain about the best algorithms that need to be implemented in order to meet the requirements. Or if efficiency is primary criteria, we are trying to let us say build software which will give very high response time, then what kind of techniques one should use in order to guarantee that kind of efficiency or response time. Now these may be some of the unknowns either to the customer or to the developer.

In such situation we might rather decide to build a simple prototype and to find out either the requirements or the techniques for implementation. So prototyping model is an intermediate model or intermediate kind of a step in trying to define the requirements or the solution approach. Usually, we will develop a prototype and we will do it in different stages. And to begin with the prototype that we will build will be based on the user needs as we currently understand them. Or the solution approaches as we understand them in the beginning and we will review and refine them.

(Refer Slide Time: 27:12)

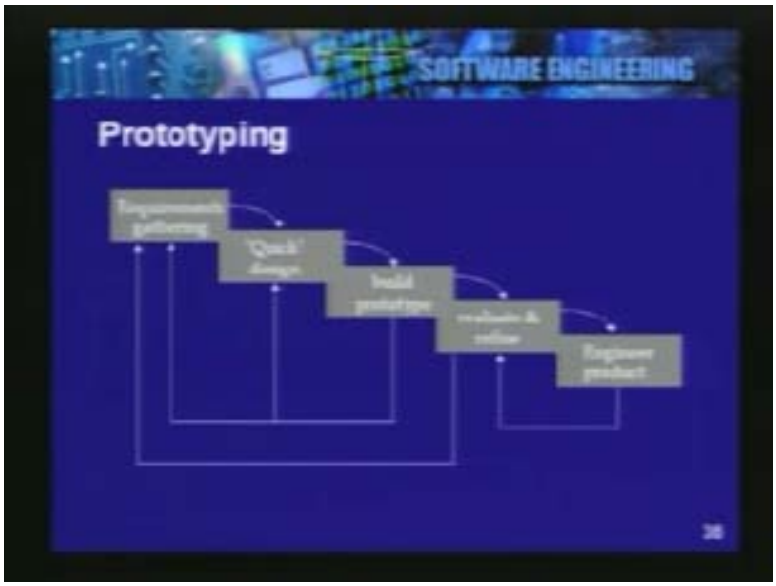


Prototype might be either a working system or it may even be a paper prototype. So idea in the case of prototype is to do a quick design and implementation and to illustrate the approach or to illustrate the functionality to the user when these are clear. So we do not want to spend too much time and effort as well as resources in building a prototype with too much detail. We want to do a quick design and in fact those features which are clearly known or clearly understood need not be implemented. In the case of prototype the focus will be on what is unknown, or not properly understood. So prototype will be tuned as we develop it. And when we try to satisfy customer needs, many iterations may be required to incorporate the changes. Every time we complete the prototype we will evaluate it, tune it and decide what needs to be done further.

Final product then would be built after the prototype is completed and is found to be acceptable. This final product may be built using the standard waterfall type of methodology, where we will go through the design, build and test life cycle. So here is the diagram which illustrates possible steps in the prototyping model. We start by quickly gathering the requirements. Then focus on those which are not clear and which need to be experimented. Then we need a quick design, we build a prototype. While building the prototype we may not care about doing too much documentation or look at the maintainability issues or even consider performance issues even if they are not relevant. Idea is to do a proof of the problem by demonstrating the unknown or un-understood.

(Refer Slide Time: 28:54)

(Refer Slide Time: 29:49)

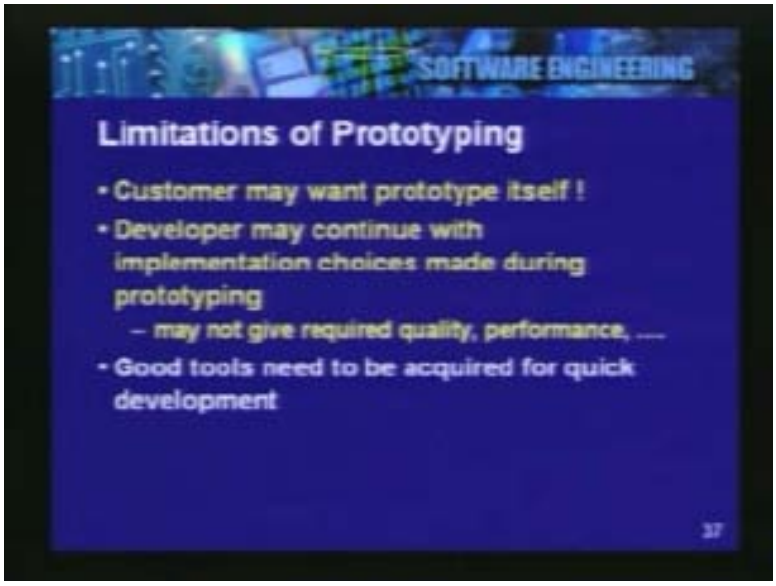


Then after we have built the prototype, we evaluate and refine. And if necessary we go back and repeat this whole cycle. When the evaluation leads to clearly defined requirements, then we are ready to build the product. And this is done using the standard engineering approach. So product development takes place once the prototyping cycles are gone through and we have come to a clearly defined requirements document for the final product. Now there are of course some limitations to prototyping. There are situations where prototyping is ideal, but at the same time we need to be aware of these limitations. First the customer may be happy with the prototype itself and may demand that the prototype be released for his operations or functions.

This is not really appropriate because the prototype may work in the beginning for some time, but it may not be maintainable. It may not give the kind of performance and it may not have the rugged design that is required for sustained operations. The second problem could be that the process of prototyping. The developer gets so attached to some of the approach he has used, that he doesn't try to refine them or look for better way of building the same functionality. In that case we may actually have difficulties with the quality performance or other attributes of the software. What it really means is that, we develop a some kind of blindness towards other possible techniques or better way of doing things, because whatever way we implemented something in the prototype, we continue the same in the product.

Good tools need to be acquired in order to do a prototype because we want to complete the prototype quickly in the minimum cost. So this is an investment we may have to make which adds to the cost of the project. So one of the problems with prototyping is that it may lead to a higher project cost and there are other problems that we have just mentioned. So when we have such difficulties based on the requirements or based on the situation we might choose some other development methodologies, one that may seem again obvious is the iterative development.

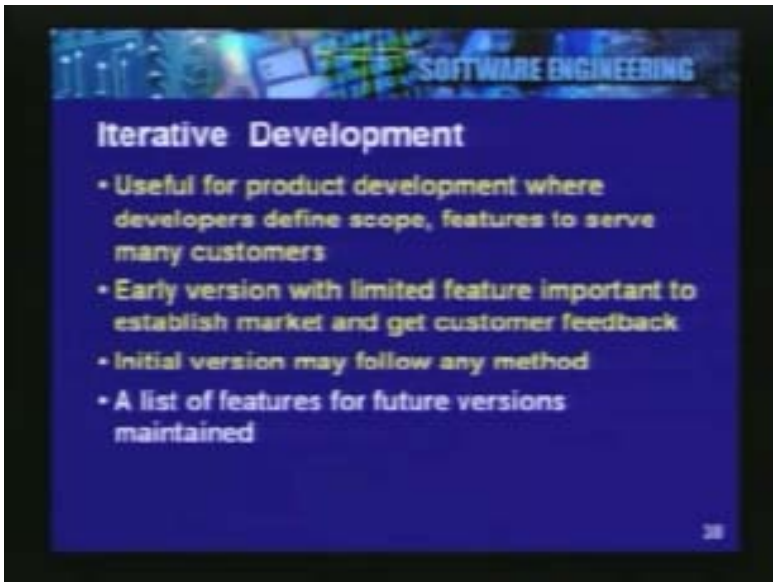
(Refer Slide Time: 32:23)



In the iterative development we define the initial scope and try to build a product which will meet the initial scope that was identified and then release it quickly to get customer feedback. Here iterative development is a good methodology for those software projects which want to build products, which will be of use to many customers. So here we are developing a product whose functionality may not be clearly known or there may be too many features that we may want to provide. But we would rather like to release a product quickly capture the market and get customer feedback. So this is the situation which is ideal for iterative development.

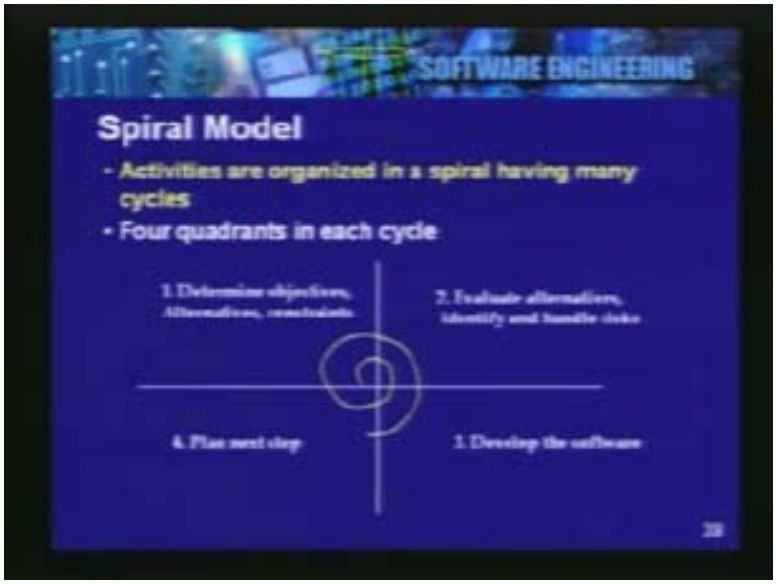
Here the development takes place in multiple iterations and many versions of the software may be released. You must be already familiar that some of the products such as database products or packages like pay roll or accounting packages are available in the market in versions. Basically the latest version represents additional features which have been added from the previous version based on the requirements as perceived or based on the customer's feedback. When we develop the initial version we may of course follow any method. For example, we may follow the waterfall method for the initial version of the product.

(Refer Slide Time: 34:53)



We generally maintain a list of features which have to be implemented in future releases and we will come out with the versions in the future as we implement these features and release the product. Each version will be analyzed and we will keep on updating our featured list. So our third methodology is the iterative development. Finally we have methodology called spiral model. In the spiral model we go through the development in cycles and each cycle is divided into four quadrants. These quadrants as you see here are named; the first quadrant, we tried to define the objectives alternatives what are the constraints. So this is the starting point. Then we evaluate those alternatives and clearly identify the risks. Now these risks may be pertaining to the interfaces, pertaining to performances and so on. Based on these risks in the third quadrant of the cycle, we tried to define what implementation will perform, and after the third in the fourth quadrant we identify how we will plan the subsequent cycle.

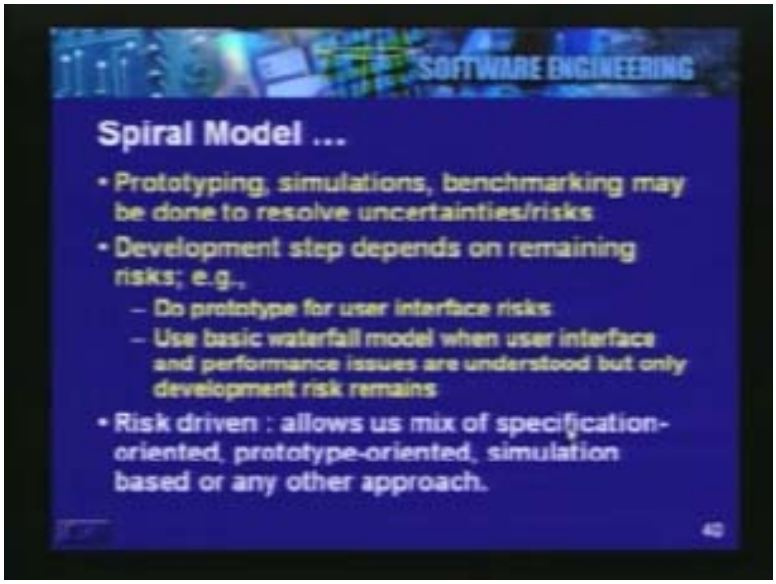
(Refer Slide Time: 36:00)



So in this case we have development going on in cycles and as we go through the cycles the different risk issues are addressed and more refinements are done to our strategies, to the functionalities in the case of spiral model. So spiral model basically identifies risks at every step and it identifies the ways of addressing those risks either by following the prototyping methods, or going through simulations, or carrying out benchmarking.

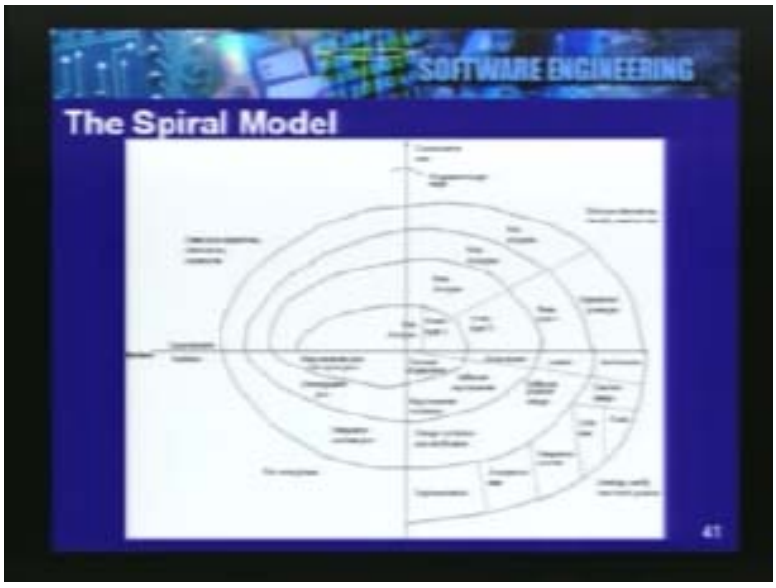
So at every development step we will decide in what way we want to follow. It is primarily a risk driven approach. The spiral model consists of different cycles. In each cycle we try to address some risk element that we see. And we plan the development, in order to identify ways of addressing that risk. So if we have a risk perception about what kind of user interfaces we should build, then we do prototyping. If a user interface and performance issues are quite well understood, then the risk may be just the development risk. In that case we decide to follow the waterfall model.

(Refer Slide Time: 37:00)



So in the case of spiral model, it's a risk driven model it allows us to use a mix of different approaches, which may be specification oriented approaches like waterfall model, or prototyping, or simulation. So unlike the incremental model where we try to release the different versions in the features of the product, here in the spiral model as we go through the different cycles, we try to address different risks. This model was proposed by Boehm and here is a diagram which defined a few cycles of the development process in the spiral model.

(Refer Slide Time: 38:28)



You can probably see it on any of the books. In each cycle the four quadrants basically try to address what we mentioned earlier. They enumerate alternatives, and then they evaluate those alternatives, try to identify the risks and build a development and carry out evaluation for the next phase. So the spiral model addresses the different risks and can be used in large projects where both the requirement risk is present and development risk is present. We have seen these different development methodologies. We saw three or four of them the waterfall model the prototype model. Now let us move on to another process which is equally important and you are going to study this in more details. But at this point i will only give an overview of the management process. As we said earlier, any large project needs to be managed. We prepare a project plan and we then manage the whole project according to that plan. So naturally the project management runs in parallel to the whole development process. As the development is progressing we are having a continuous management oversight.

So in the case of project management, we have primarily, activity of defining the project plan. In the project plan we will identify different activities, resources, deliverables, and timelines and so on. So once the plan is ready then we will monitor it and also control as it progresses. A project plan indicates how the project will be executed. And there are very important things we must understand about the planning. A project plan is absolutely essential for successful completion of the project. Therefore we must appreciate some of the points which are important in a project plan. The first is that if you do not have a plan then there is no management. There is no question of doing any management if the plan does not exist. Then without making any measurements, it is not possible to prepare any plans.

(Refer Slide Time: 40:35)

This is a very important truth that in order to prepare a plan, you must identify the activities not only qualitatively but quantitatively.

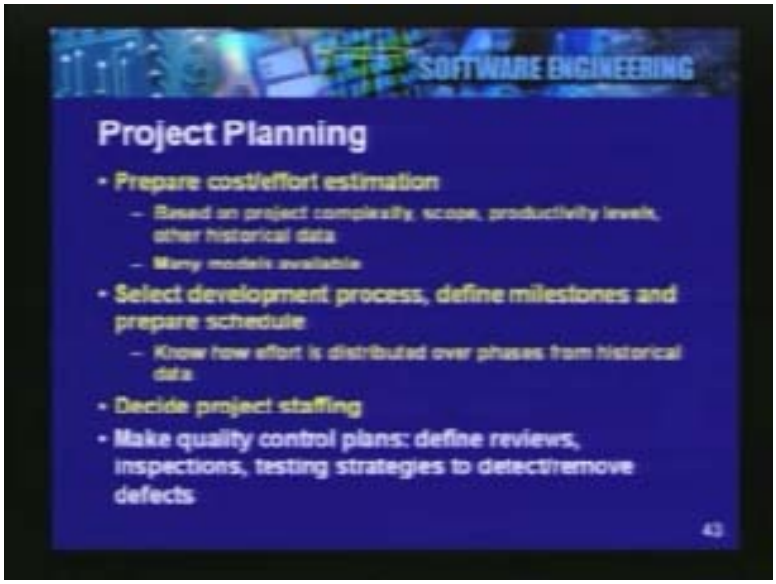
You must learn to measure those aspects which we want to bring under the management control. So without measurement, planning is not possible. Without planning, management is not possible. It is very important that first we learn to make measurements; we measure those aspects of the project which we want to bring under the control of the management. Make a plan and then perform the management as we continue with the development. That plan allows progress to be measured. Since it consists of different activities, as we complete these activities, and as we know the effort distribution among these activities, we know how far we have progressed and how much still remains to be done. So when you have the plan you can measure the progress.

We generally prepare this plan before the development begins. As we said earlier it may be produced along with the analysis or even before the analysis is completed. So we prepare a plan as soon as we have a good idea about the complexity of the project and we start getting a good feel of the requirements. Once we have done this, obviously this plan needs to be constantly updated. We will get management inputs from the development team. And based on the progress of the work, we may have to update the plans we have already made. How do we prepare the project plan? What are the important steps that need to be taken in this? The first important step is that we must estimate the cost and effort. This is the quantification that we talked about just now.

We need to measure how much effort is involved in carrying out a project. Now this will naturally be based on the complexity of the project, the scope of the project, productivity levels of our people, who will be implementing the project, and also other historical data of similar projects. We will naturally benefit from the previous experiences. So we should clearly keep a record of other projects performed by us. Understand our productivity levels and then estimate the effort or the cost for the new projects that we are undertaking. There are different models available for cost and effort estimation. And in this course later on you will learn more about these models. Once we have done the effort estimation, then we select the right development process.

We said earlier that there can be many different types of processes. So you choose the one which is best suited for task at hand. You define the milestones and you prepare a schedule. In order to prepare this schedule we of course need to know how the effort is distributed over the different steps. We had seen one graph where we had shown the relative effort distribution over different steps in a typical waterfall kind of a methodology. We should have our own data from our past experiences. So that the overall effort estimated by us can be distributed over these different steps and we can prepare a detailed schedule for execution of the project. And then we decide on project staffing; who will perform the different steps, how many people will be made available, what are the scheduling requirements, what are the overall calendar time in which the project needs to be completed. We will also make explicit quality control plans. In the overall plan, we will schedule reviews and inspections at appropriate points.

(Refer Slide Time: 45:42)



We will define testing strategies, so that we can maintain a control over the defects and we can ensure a quality product from our approach. These are the different steps in the preparation of a project plan. You need to identify the effort. You need to know how the effort is being distributed, you need to commit resources for different steps and you need to make explicit quality plans. Because quality will naturally have cost implications, you will need to spend effort in identifying the defects, which might go from one step to another through the various deliverables that the steps produce. So these are the different project plan elements.

Once the project plan is in place we will use it for monitoring and control. The plan defines the various activities we may represent it in different ways. There are Gantt chart, Time bar chart or PERT by CPM type of activity networks which are used for representing a plan. Essentially the plan shows the different activities. It shows expected durations for different steps and what kinds of resources have been allocated to those activities. In a network of activities like this, we can identify which is the critical path. If we do not meet the schedule requirements for the critical path then the project may get delayed. So we identify which are the critical activities and how to keep them under control. So that they are carried out in the expected duration and in the expected cost. If it is not done then there is going to be a significant impact on the overall project.

(Refer Slide Time: 47:05)

So these are important project monitoring aspect and we use different types of analysis techniques by representing the project plan in a proper form such as a bar chart or as activity network. Each development step you will recall, will give information for tracking the progress of the development. We had said earlier that every step produces input for the management and this information will be used by the management for identifying possible bottom necks and the reason for delays if there is any. And we will enable the management to take corrective action. So project plan allows you to monitor the project and also control over the progress of the project. Let us now summarize we have in this particular lesson.

We have we have introduced the software engineering topic. We have looked at the importance of applying engineering approach to development of large software project. We define what we mean by engineering. So now we should be very clear how we benefit by applying an engineering approach to software development. We identified different types of software processes. We defined a stepwise process definition where the software methodology consists of multiple steps. We made a very rigorous definition of what we mean by a software development step. It has entry criteria exit criteria something specific to be done, some techniques to be employed some documentation, or some other output to be produced which will be used by subsequent steps. So the essence of engineering approach is to define a stepwise methodology for carrying out the project.

We saw the waterfall model and other models. We tried to understand their strengths and their weaknesses. We saw that the waterfall model consists of many steps arranged in a proper sequence and it identifies many different deliverables which are very important not only for completing the project but for future maintenance of the project. We saw then limitations of waterfall model and how other models tried to address those limitations such as the prototyping model or the spiral model.

And finally we briefly talked about the project planning. Key observations we made here is that if you do not have a project plan you cannot manage the project. And project management is actually very important in any large project. And another key point was that, in order to prepare a project plan it is important that we learn measurement techniques. We keep historical data, we learned how to effort how to do the effort estimation. We also understand how the effort is distributed in different steps. So once we have a good understanding about these issues and we also maintain data about similar projects that we might have done in the past, we will be able to prepare the project plan with a great amount of confidence and we will be able to carry out the whole project as per the plan.