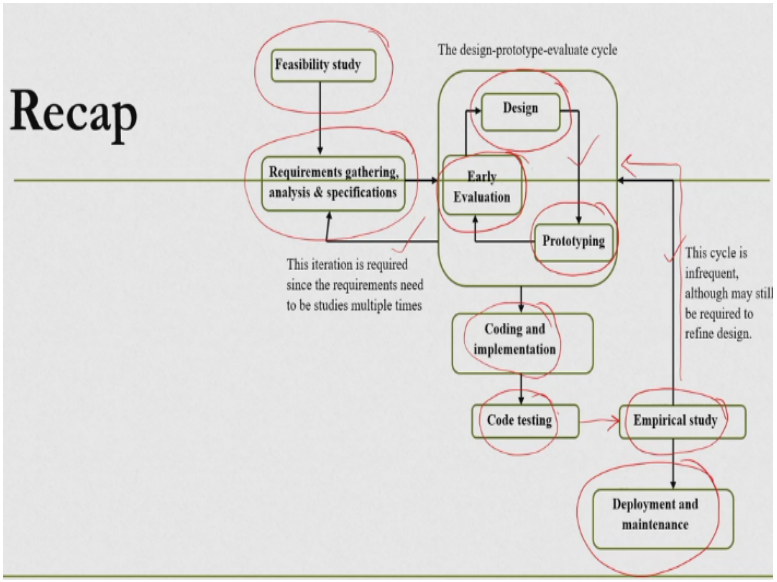
Design and Implementation of Human-Computer Interfaces
Dr. Samit Bhattacharya
Department of Computer Science and Engineering
Indian Institute of Technology – Guwahati

Lecture – 42
Project Management

(Refer Slide Time: 01:00)



Hello and welcome to the NPTEL MOOCS course on design and implementation of human-computer interfaces. We are going to start lecture number 36 where we will talk in brief about the concepts of project management. Before we start, let us quickly recap what we have covered so far. We are discussing the interactive system development lifecycle. This lifecycle is required to develop and deploy human computer interfaces, which are examples of interactive systems.

Just to quickly recap why this lifecycle is important, so in interactive systems or more specifically in the context of human-computer interfaces what we need is basically a system that is executable and at the same time usable. So, we have discussed the idea of usability in several lectures and we need to ensure that the system that we develop is usable as well as executable. In order to achieve that, we need to follow some systematic development approach.

Lifecycle development models that are typically followed in software engineering provide such systematic approaches. So, there are several models. For example, we have seen the

water-fall model, the spiral model. Spiral model is of course a meta level model and it involves everything whereas water-fall model is not suitable for use in the development of interactive systems.

In interactive system development what we require is involvement of the end users, not the clients or customers, rather the end users in the development process either as active user or passive user. In other words, what we are trying to do is to follow what is called user centred design approach. Now, that involves lots of feedback from the users which needs to be incorporated in the development process which in turn implies that our lifecycle should have iterations between stages.

So, the characteristic of any interactive system development lifecycle is the presence of iterations and incorporation of user feedback through these iterations in the end product. In this course, we are learning such an interactive system development lifecycle model. In our model, there are several stages which we have covered. So, there is the feasibility study stage which we ignored. This is of course very important and should happen at the beginning of any development process.

We started our discussion with the requirement gathering analysis and specification stage. Next is the design, prototyping and evaluation cycle comprising of these three stages; the design, prototyping and evaluation. Now, this cycle is meant to take into account the usability concerns. So, design refers to design of interfaces and interactions. These designs are typically done based on guidelines to ensure that the interfaces and interactions that are designed are usable.

However, to test whether they are indeed usable or not, those are prototyped and evaluated with experts. If some problems are found, then the designs are refined, again prototype, again evaluated and this goes on in a cycle design, prototype, evaluate cycle. Note here that we are taking care of the iteration here. We are taking care of the user feedback through the iteration here. Now, of course in this evaluation, we are not actually involving the end users actively.

Instead, we are using representation of such users in terms of domain experts who are supposed to represent the end user behaviour. So it is kind of we can think as passive involvement of end users. This is followed by design of the system, which is not shown

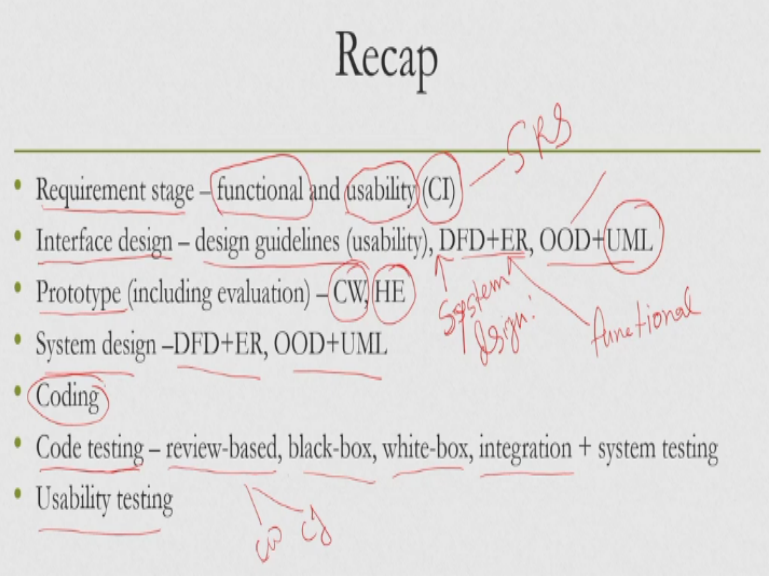
separately in this lifecycle model. So, once we arrive at a stable interface design, we go for designing the system in terms of modules and submodules and their connections that is followed by coding and implementation of the design system followed by testing of the code.

This is followed by, one is missing here, this is followed by a phase called empirical study in this phase which we have covered in details in the previous few lectures. We test the usability of the system with actual end users. So, input testing we test the execute ability of the system, in empirical study we test the usability of the system. Now it may so happen that at this stage we may find some usability issues, which we can fix by going back to the design stage.

There we may or may not follow this design, prototype, evaluate cycle again depending on the availability of time and resources. However, we may refine our system design, refine the code, test the newly added code part and then we may conduct another empirical study to check whether the system is usable. Now, here also there is one cycle as you can see from empirical study it goes back to the design stage. So, at multiple places we have these cycles. Another cycle exists between the design stage and requirements specification stage.

So, in the design stage if we find too many problems, then we may need to revise our requirements and then go back to the design. So, three places we have cycles or iterations between the requirement and design phase; in the design, prototype, evaluate cycle and between the empirical study and design phase. Finally, we come to the deployment and maintenance stage. So, these are the stages and the systematic approach to develop interactive system.

(Refer Slide Time: 07:24)



Now, what do we do in these stages? Let us have a quick look. In the requirements stage, we gather requirements at two levels, from the client or customer as well as the end users. Traditionally in software development, client requirements are gathered whereas in case of interactive system development along with client requirements, we also need to gather end user requirements.

So, typically client requirements can be represented in terms of functionality requirements, plus there will be some non-functional requirements that are part of client requirements as well. Whereas when we gather requirements pertaining to end users, we call it usability requirements. Now, we have discussed one such technique, there are several techniques available for collecting end user requirements.

We have discussed one such technique which is called contextual inquiry. This is an observation-based requirement gathering technique if you may recollect, where we observe the user behaviour and record those behaviour for analysis. At the end, what we get his SRS document which is our starting point for system design. In the interface design phase, what we do? We use design guidelines as mentioned earlier for ensuring usability.

Now, a couple of such guidelines we have studied; Shneiderman's eight golden rules, Norman's seven principles all these things we have studied. In the system design phase, what we do? So, we should treat them separately, one is interface design, one is system design. We come up with design of the system. Now, we can follow either of the approaches. If we

follow the functional approach then generally, we use DFD or data flow diagram to represent our design.

Alternatively, we can use object-oriented approach or object-oriented design approach OOD, there we can use the UML notation Unified Modelling Language notation to represent the design. Now, it may be recalled here that in system design our primary focus is hierarchical design, where we divide the problem into modules and submodules so that it can be distributed to a team for parallel development.

And also it is very important to define the interfaces between the different modules. Now, in the prototype phase that is related to the interface design as well as the evaluation phase, we develop prototypes. So, we learned about several techniques to build prototypes, high fidelity, medium fidelity, low fidelity as well as vertical, horizontal. Also, you have learned about couple of prototype evaluation techniques such as cognitive walkthrough and heuristic evaluation.

So, these are used for evaluating interface design and refine the design if required, but these are not relevant for system design. In system design as I mentioned, so use functional or object oriented and correspondingly we use the languages such as DFD or UML for representation. This is followed by coding stage. In coding what do we do? We write codes to implement the system.

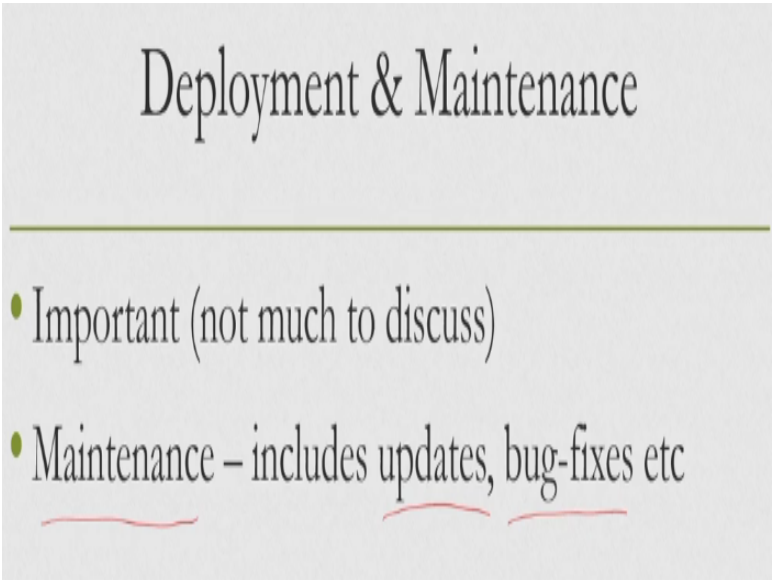
So, good coding practices are the primary concern here, we should follow those to write code that is understandable and maintainable. This is followed by code testing, we try to find out whether there are any issues with the code that we have written. Now, code testing there are several methods, we have learned about those. A quick testing method is the review-based methods. Here we have learned about code walkthrough method and code inspection methods.

These are similar in nature to cognitive walkthrough and heuristic evaluation methods as we have discussed during the lecture. This is followed by a more systematic method namely the functional testing or the black box testing as well as the structural testing or white box testing. Now, these testing are primarily done at the unit level at the module level. So, we divide the code system into modules, implement them and test them using these methods.

It is also necessary to perform testing of the system after integration of the modules, so that is also important and we have seen several such testing methods. Finally, we discussed in the previous few lectures about another aspect of system testing that is usability testing through empirical method where we have seen that we follow a very systematic and rigorous method comprising of four steps; research question formulation, variable identification, design of experiment and data analysis.

In data analysis, we primarily test for statistical significance of the data that we observed to draw a reliable conclusion about the usability of the end product. So, these are in summary the things that we have already covered in our previous lectures.

(Refer Slide Time: 13:32)

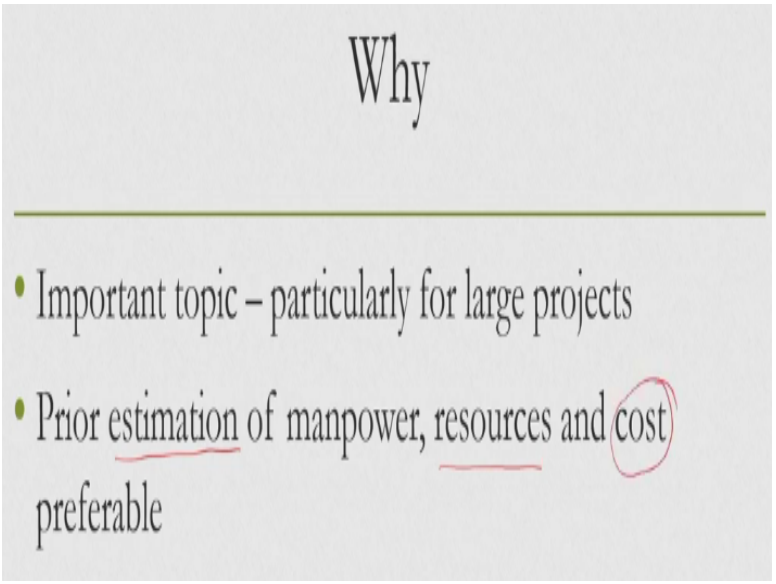


One stage that remains to be covered is the final stage that is deployment and maintenance. Of course, this is very important, but in this course we are not going to discuss much about it that is not the focus of this course. It will be sufficient to know here that in the maintenance stage, what we do is we need to keep track of issues with the system that we have already deployed in the market and we need to provide updates to the clients, fix bugs if some are reported by the clients and so on.

These are typically part of the maintenance stage. What we are going to focus next is a topic that is very important and typically that is done at the beginning of the development process, although we are discussing it at the end, but typically you should keep in mind that this is supposed to be done at the beginning before the full development takes place that is project

management. Let us see what are the issues in project management and how we can address those issues.

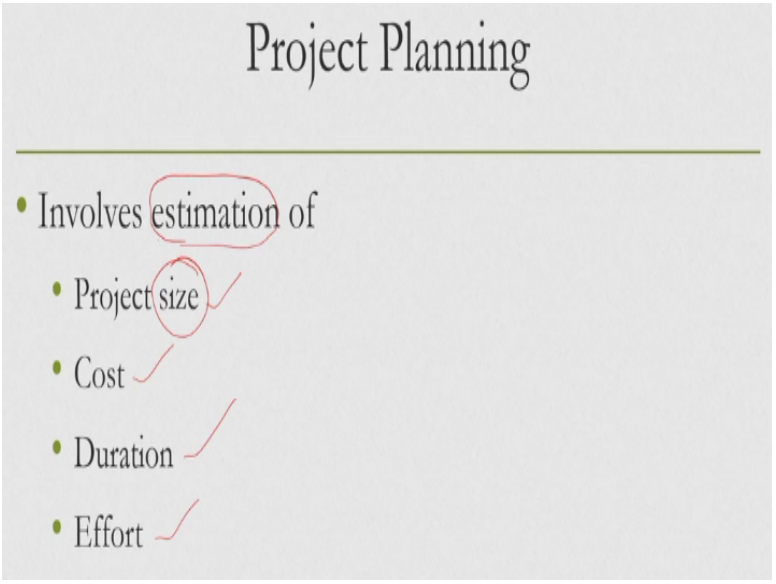
(Refer Slide Time: 14:32)



Now, project management is a very important issue when we are dealing with practical large and complex systems. So, they are if we simply start development process without actually properly planning for different stages of execution as well as allocation of resources, then it may end up in complete chaos and we may miss the deadlines. So, it is very important to do it before we start the process. Now, what it involves?

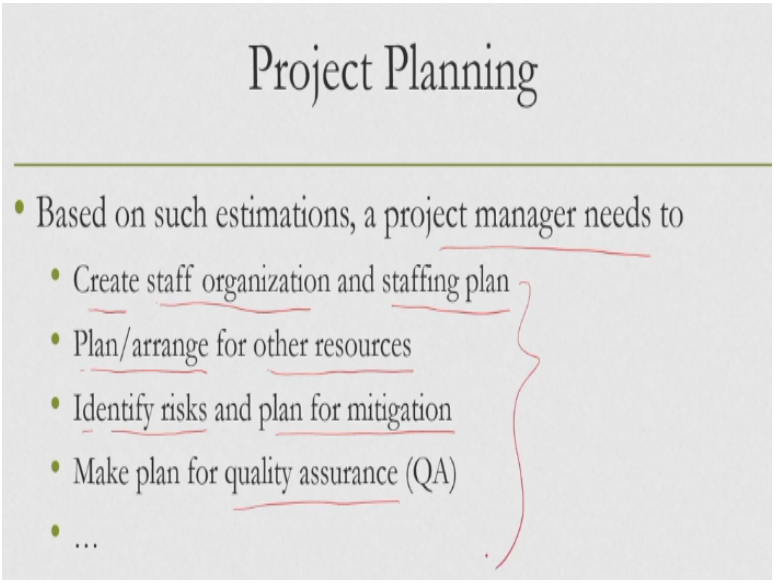
It involves prior estimation of the manpower that may be required, resources that may be required and the cost that is going to be incurred before the development is complete and the system can be deployed. So, typically these three things we are interested in; the manpower requirement, the resource requirement as well as the overall cost. So, a project planning in order to answer these questions of manpower requirement, resource requirement or cost we need to properly plan the project.

(Refer Slide Time: 15:47)



Now, planning involves some estimation. Prior estimation before the actual things take place prior estimation of the size of the project, cost, duration and effort that is required to be put in. So, these are key considerations while we go for planning of a project.

(Refer Slide Time: 16:05)



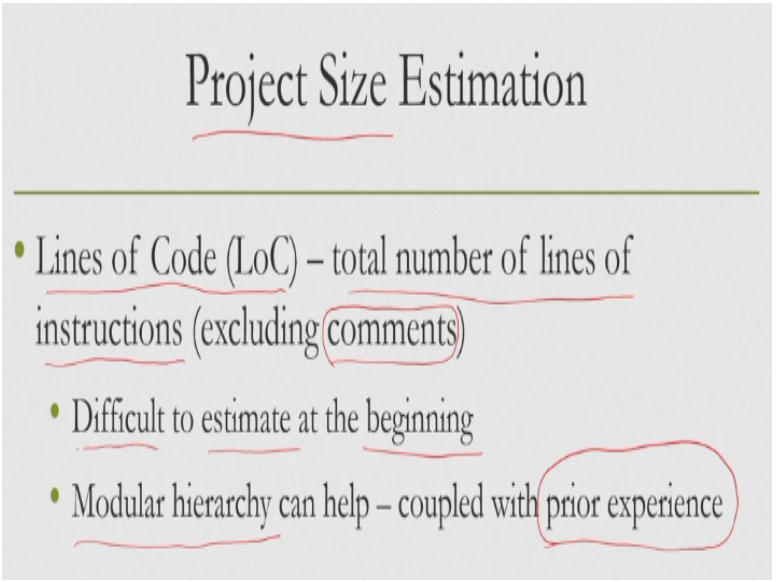
Once we are able to make such estimates assuming that we can do that, once the estimates are made, then what a project manager needs to do? A project manager needs to create staff organization and staffing plan, so how to allocate staffs, how to hire staffs. Plan or arrange for other resources, hardware, software, infrastructure, physical infrastructure such as seating arrangements, electricity, etc. Identify risks and plan for mitigation of those risks.

So, while the development is taking place, what kind of issues that may crop up, what are the risks involved and how to mitigate those in case of those risks are encountered during the

development phase. Also, the manager needs to make plan for quality assurance, quality of the end product that needs to be assured. So, these are some of the key considerations, there are other considerations as well.

So, once some estimate is made based on the project size, cost, duration, effort, then the manager can plan for these activities how to hire staff, then resources required, then the risks that may happen, risks associated with the development and how to mitigate if those things happen, quality assurance how to take care of that. All these issues need to be taken care of by the project manager. So, the job of the manager is not easy.

(Refer Slide Time: 17:51)



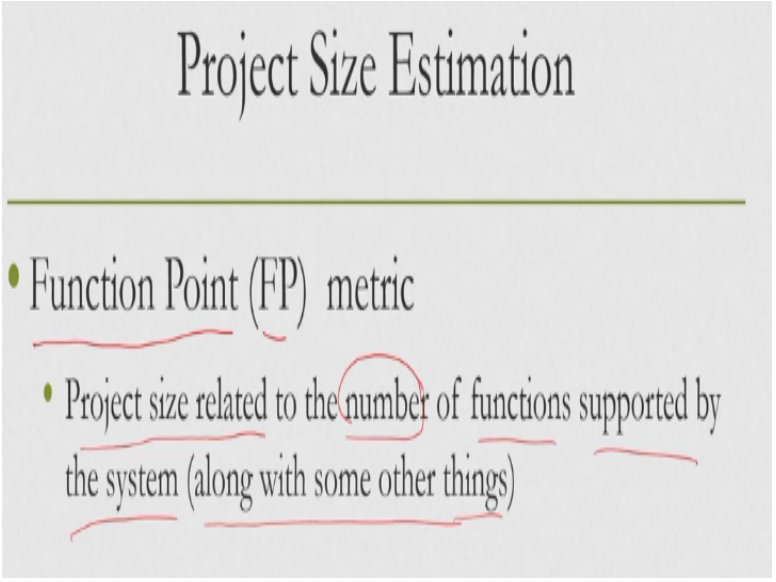
Now, let us have a brief idea about these different estimates that we need to make. The most crucial estimate is project size. So, how do we estimate the size of a project? Now, here we can simply estimate the size in terms of the size of the system. So, essentially a project is nothing but development of a system, so the size of the system we can use as the size of the project. Now, the size of the system, so here we are talking of software systems.

So size of the system can be simply the size of the software that is the lines of code or Lo C that is the total number of lines of instructions that is part of the system. Of course, this consideration excludes any line that is used as comments. So, we need to focus only on the lines of instructions rather than comments. So, line of code is one very obvious and intuitive way of estimating the size of a project. But there are issues involved.

Most importantly, it is very difficult to estimate at the beginning of the development what is going to be the size of the project. That is of course as you can understand is very difficult to estimate at the beginning unless you have some experience of developing similar systems. Another way to overcome it is to make use of the modular hierarchy. So, this requires you to first design the system in terms of modules and then go for estimation, so anyway this cannot be done at the very beginning.

It has to be done after some progress is made in the development process, namely requirements are gathered and we have arrived at the design stage. If we want to avoid that, then probably we have to rely on prior experience. In fact, in the modular hierarchy also, we can have prior experience for similar systems, how many modules are typically part of such systems and based on that experience also some estimates can be made. So, all depends on primarily the experience of the manager.

(Refer Slide Time: 20:16)



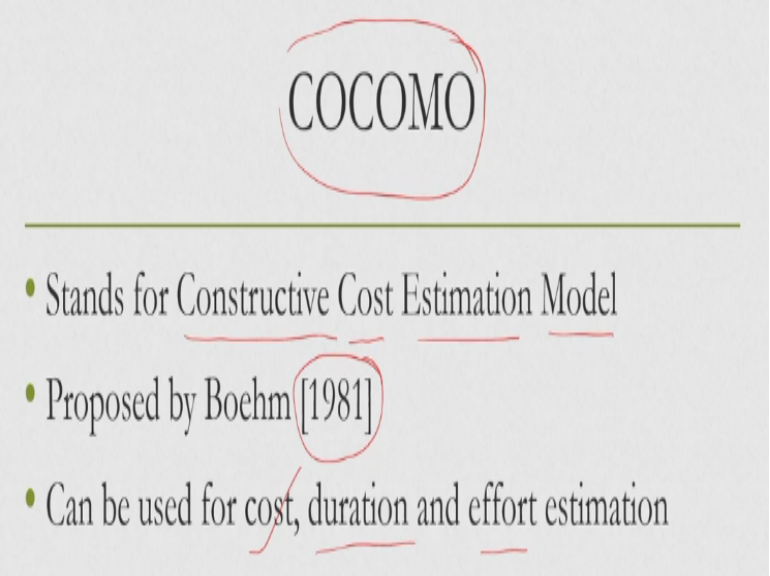
Another metric for estimation of project size is function point or FP. So, it tells that project size is related to the number of functions which is supported by the system. So, essentially what it tells is that the system or the software is nothing but a collection of functions. So, the total number of functions that is going to be part of the software can be used to estimate the project size and we call this function point metric.

Of course, there are some other considerations as well, but those we can ignore for simplicity and say that function point metrics refers to the number of functions that is part of the software. Again, here some prior experience is required to estimate this number before the

system is developed that is about size estimation. So, we discussed two concepts; one is line of code, other one is function point.

There are many other ways to estimate size, those are more complex. So, we are not going to deeper into those concepts to keep it simple.

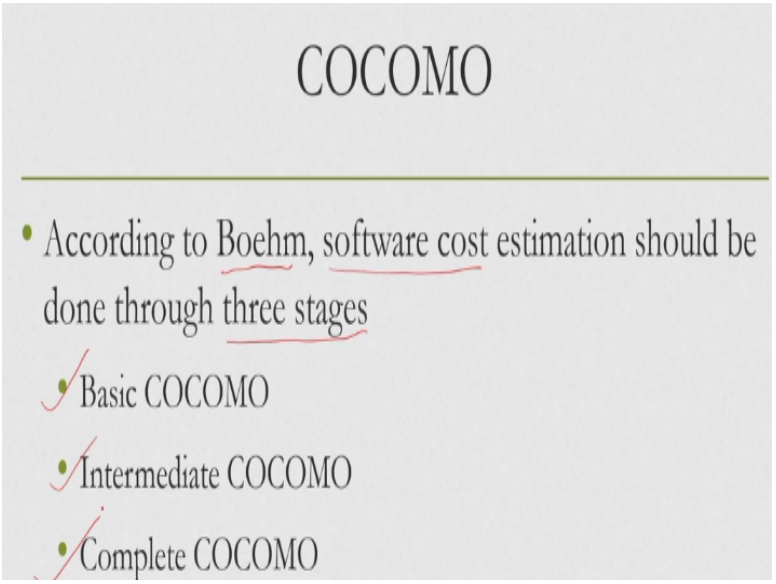
(Refer Slide Time: 21:39)



Next is the estimation of cost. There is a very famous model, although very old, that is called COCOMO which stands for constructive cost estimation model. Now, this model was developed way back in 1981. It was proposed by Boehm say about 40 years ago. And this model can be used to estimate the cost of a project. It can also be used to estimate the duration and effort required for developing a system along with the cost estimates.

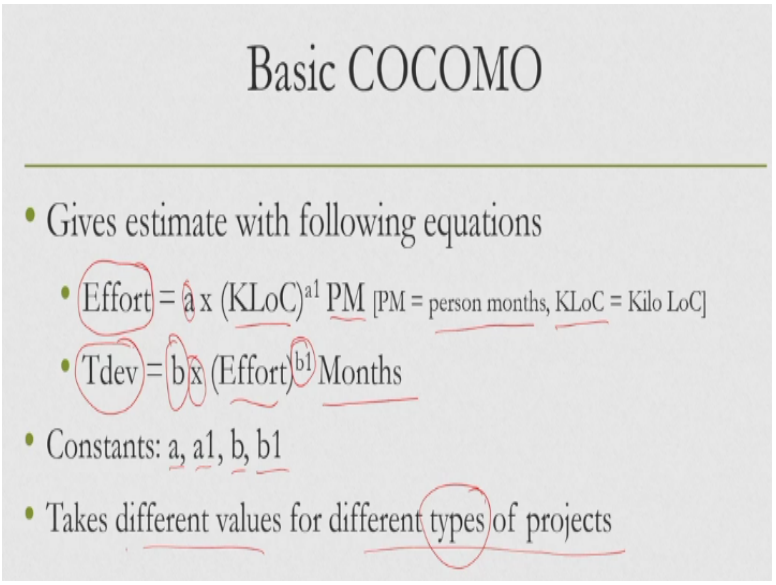
So, earlier we have seen size estimation, with COCOMO we can estimate the cost, duration and effort, the other three primary concerns of a project manager.

(Refer Slide Time: 22:31)



Now, the COCOMO is not a single model. According to Boehm, software cost estimation should be done through three stages and accordingly three different models were proposed. One is the basic COCOMO model, one is the intermediate COCOMO model and other one is the complete COCOMO model.

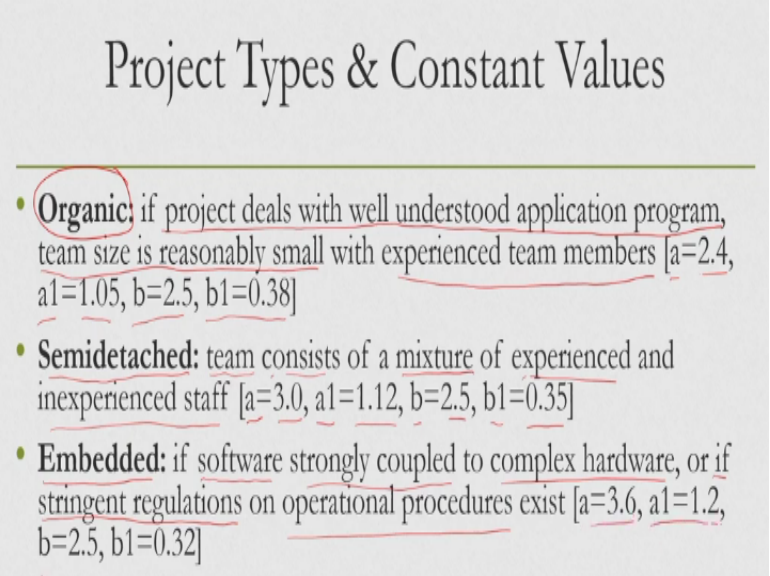
(Refer Slide Time: 22:58)



The basic COCOMO model gives estimate with an equation that is effort is expressed or can be computed in this way a into K LoC to the power a 1 PM, now here PM stands for person months and K LoC stands for kilo lines of code or 1000 lines of code. Another metric is T dev time to develop which is b into effort to the power b1 and the unit is months. So, the unit for effort is person months and the unit for total development time is months.

Now, in these equations a, a 1, b, b 1 these are constants and these takes different values for different types of projects. So, you can categorize projects into different types and depending on the types these constants take different values. Those values were estimated empirically after studying several projects. Let us see what kind of values they take for different types of projects.

(Refer Slide Time: 24:20)



One project type is organic. Now, what this project talks about is that if a project deals with a well understood application program and the team size is reasonably small with only experienced team members. Then we call those projects belonging to organic types. So, to repeat there is one category of projects called organic type of projects, these are projects that are basically dealing with well understood application areas that involve team size small and all the team members are experienced in similar projects.

So, if that is the case and we are dealing with such a project, then the constant values a, b and a 1, b 1 takes these values; a = 2.4, a 1 = 1.05, b = 2.5 and b 1 = 0.38. So, with this set of values what we can do is as we can see once we know these values, so for organic projects we know the values, then we can replace these a, a 1, b, b 1 in these equations and with our estimate of lines of codes.

We can first calculate effort by replacing a and a 1 with those values then effort is calculated and with the effort value and replacing b and b 1 with those values for organic project we can get development time. So, these will help us in estimating the effort and time. Another category is semidetached. So, here we consider the project to belong to this category if the

team that is going to develop the project consists of a mixture of experienced and inexperienced staff.

So, earlier we talked of a team where only experienced staff are part of the team, whereas in semidetached projects the team comprises of experienced and inexperienced staffs. So, in that case we have a = 3.0, a 1 = 1.12, b = 2.5 and b 1 = 0.35 that is when we are considering semidetached projects. Finally, we have embedded projects. Here a project belongs to this category if the corresponding software that we are developing strongly coupled to some complex hardware.

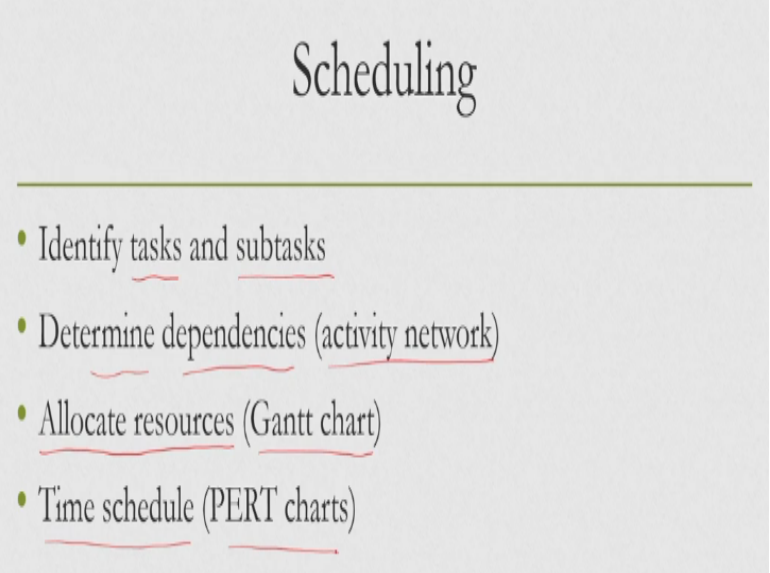
So, the software has some associated hardware attached to it, so we are specifically building for that hardware or if there are stringent regulations on operational procedures for developing a software, some constraints are there. So, if we are dealing with such a project where we are asked to build a software that is coupled to a specific hardware or some specific constraints are there then we call this project to belong to this embedded category.

In that case is different value 3.6, a 1 is 1.2, b is 2.5 and b 1 is 0.32. So, broadly there are these three categories organic, semidetached and embedded. And depending on the category of the project, we can choose the values of a, a 1, b, b 1 these constants. We can use these values in the basic COCOMO model to compute the effort and development time which are basically estimates and then based on that we can make project related decisions.

One thing that we skipped here is that although we mentioned that there are three models basic, intermediate and complete COCOMO, we briefly discussed only the basic COCOMO model for simplicity. Now, total cost estimation involves estimating these three stages and then going for cost estimation. So, we only discussed basic COCOMO, we skipped the other parts for simplicity. Next is scheduling.

So far we talked about the major concepts that is size estimation. So, we saw that we can do that using lines of codes or function points. Then cost estimation, we have seen that we can use the COCOMO model. The model also gives us estimates enough effort and development time. Another important job of a project manager is to go for scheduling.

(Refer Slide Time: 29:44)

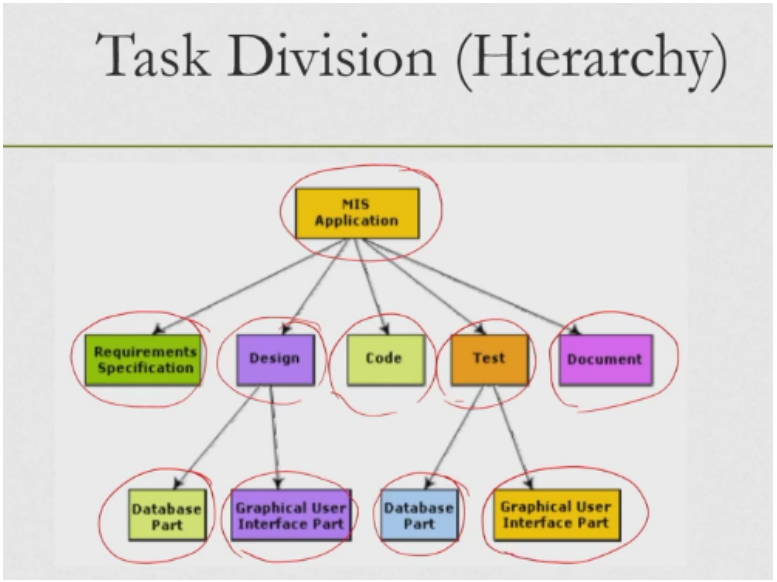


Now scheduling involves identification of tasks and subtasks to be done for the development of the system and then scheduling those tasks and subtasks in appropriate manner. So, after this identification, the manager needs to determine the dependencies between those tasks and subtasks, how one task is dependent on others. So, this can be done with the help of a graphical language that is called activity network.

Once those dependencies are known, then the manager can allocate the resources to carry out those tasks and this can be represented with a Gantt chart. And finally, a time schedule needs to be prepared which can be represented using a PERT chart. So, all these activities need to be performed to properly schedule and represent that schedule. So, let us have a quick understanding of these concept; activity network, Gantt chart and PERT chart.

First thing is we need to identify we as a project manager. If we are project managers, then we need to first identify what are the tasks and subtasks required to complete the development process and we need to create some sort of hierarchy for better understanding.

(Refer Slide Time: 31:19)

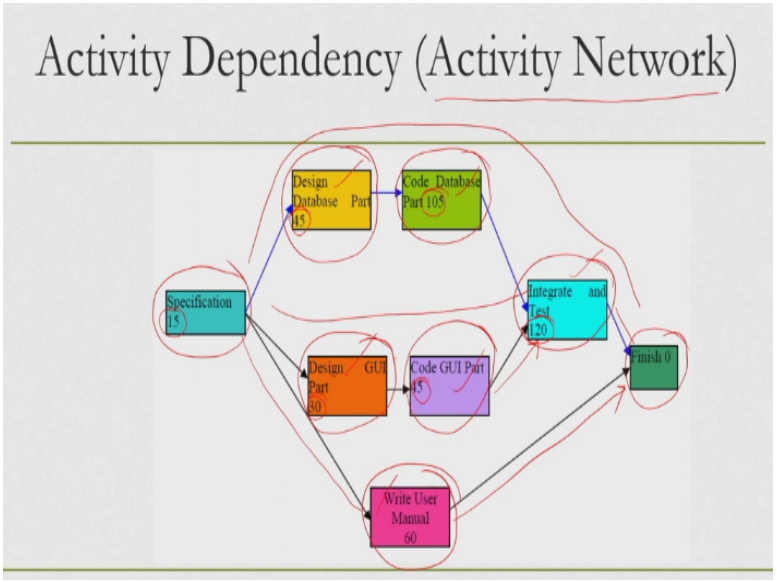


So, one example scenario we are considering here that is one specific application which belongs to broadly the MIS application category, Management Information System application category. Now, here as you can see there are several tasks identified. Requirements specification is a task, design is another task, code development is another task, testing is another task, document creation is another task.

Now under design, there can be subtasks like database part design, graphical user interface design. Similarly under test, there can be database part testing, graphical user interface part testing. Roughly it follows our lifecycle of course, but this is a simple example of identifying tasks and dividing it and organizing it into hierarchical form.

So, that is the first thing that a project manager needs to do, identify the tasks and come up with some sort of hierarchy of these tasks for quick understand. Next thing is the manager needs to then find out dependencies between these tasks. Another example is shown here.

(Refer Slide Time: 32:29)



So, this shows a particular notation which comes from the presentation scheme called activity network. So, here suppose there are several tasks. So, specification part takes some time, mentioned here, we can consider it as to be weeks or months or days anything or hours up to us. So, a specification involves some time. Then from specification there is this design database part which again involves some time.

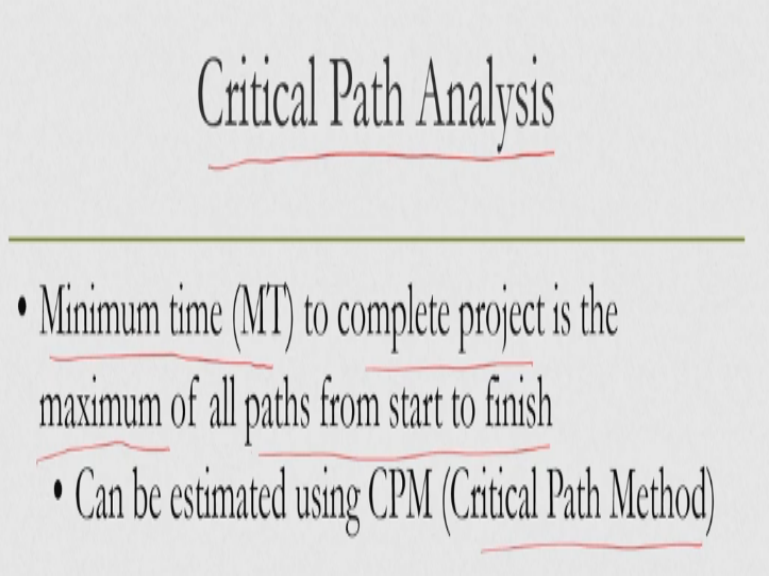
Coding of the design database part involves some time, then integration and testing with time mentioned how long it should take, finally reaches the finish stage that is one path, this path. There are two other paths, from specification it goes to design GUI part with some time mentioned say 30 days, then could code GUI part say 45 days it takes, from there it goes to the integrate and test part, so this is another path and from there it goes to finish.

Alternatively, from specification it can go to write user manual part 60 days suppose and then go to finish part. So, one path indicates that design and testing of database part, another path indicates design and testing of GUI part and third path indicates creation of document that path. So, what this network shows us? It shows that we need to follow these three paths in these orders. We cannot follow the other orders like first we create the code and then go for design.

We cannot follow that or we cannot start with this integrate and test part before we finished these other parts like design database, design GUI, write manual and code database, code GUI. Unless we do these things, we cannot do the integrate and test part. So, this shows the dependencies, what is done after what or what precedes which activity or task. So, this is a

graphical way of representing the dependencies between tasks that have been identified in the earlier stage.

(Refer Slide Time: 35:22)

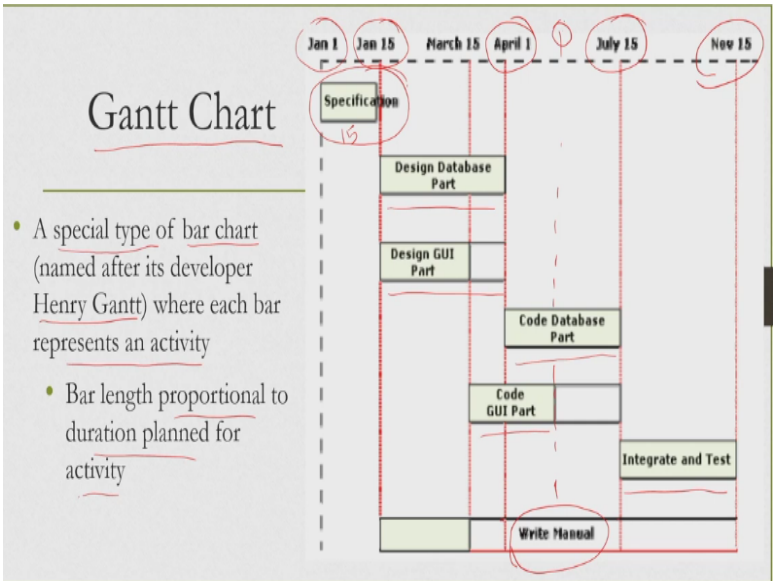


Also, from this activity network we can do some more things. One important thing that we can do is called critical path analysis. What it does? It tells us the minimum time required to complete a project. Now, we can compute it from the time units that are mentioned for each task. Of course, those are our estimates and from those estimates we can compute the minimum time required to complete the whole project incorporating all the components of the project and that is called critical path analysis.

So, how we can do that? By finding the maximum of all paths from start to finish in the activity network. Now, this can be done by using the critical path method or CPM. So, we will again skip, we will not go into the details of this method, but that is what we can do if we are able to represent our dependencies between tasks in the form of activity network with time estimates mentioned for each task.

If that is the case, if we are able to do that, then we will be able to basically find out the critical path that is the minimum time required to complete the project by following the critical path method.

(Refer Slide Time: 37:06)



Then we need to allocate resources and for that also we can use one graphical notation that is called Gantt chart. So, this is a special type of bar chart which is named after its developer Henry Gantt where each bar represents an activity and the bar length is proportional to the duration planned for activity. For example, here we can show a Gantt chart of course shows a kind of schedule so as you can see specification is shown in the form of a bar, horizontal bar and on top you can see the dates are mentioned.

So, as you can see here in this example specification task is represented with this horizontal bar, it is to be completed between January 1 and January 15, so 15 days of time is given. Now you know how we get this time in the activity network, so we can schedule it in this way. Then between January 15 to April 1, the design database part as well as the design GUI part needs to be completed, so that is the total duration.

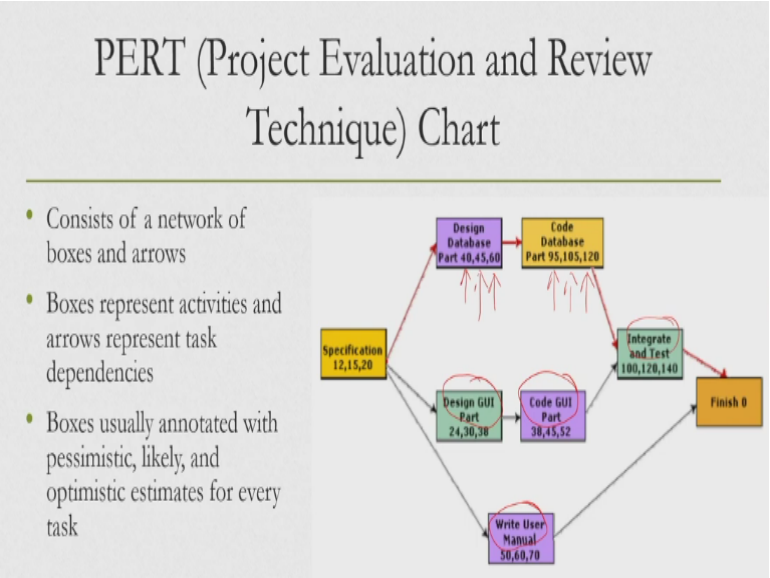
Then from April 1 to July 15, the code database part needs to be completed whereas code GUI part is supposed to be completed between March 15 to somewhere here as shown with this line. So, no date specifically is mentioned just for simplicity, but you are supposed to put a date here. Integrate and test this activity needs to be done between July 15 to November 15. And between April 1 to July 15, manual writing activity needs to be done.

So, in this way it can be sewn how the different tasks can be scheduled and accordingly you can plan for allocating resources to those tasks so that those tasks can be completed on schedule. And the length of the bar is of course as it is obvious proportional to the time it

takes to complete the activity. So, the longer it takes the length will be larger. So, under scheduling, we talked about identify tasks and subtasks that we have covered.

Then for determining dependencies, we first need to create an activity network and from there we can see that dependencies. To allocate resources we need to create a Gantt chart and based on the schedule that we have created there, the times we can allocate resources so that those activities get over within that scheduled time within the deadline. The last thing that remains is the time schedule. This we can achieve with a PERT chart. So, let us see what is a PERT chart.

(Refer Slide Time: 40:58)



Now PERT stands for Project Evaluation and Review Technique, so this P E R and T together produces this acronym PERT. So, this is another graphical notation to represent the time schedule. Now, it consists of a network of boxes and arrows somewhat similar to the activity network, but with some additional information. So, the boxes represent activities and the arrows represent task dependencies, we have seen earlier.

The boxes are usually annotated with pessimistic, likely and optimistic estimates. So, there are three estimates now, instead of one which was there in the activity network. So these three estimates stand for pessimistic estimate, likely estimate and optimistic estimates for every task. For example, specification earlier we mentioned only one estimate which was 15 days as we can see from the Gantt chart.

But now we also had a pessimistic estimate and optimistic estimate, so optimistic estimate is 12 that is it will take less time and pessimistic estimate is 20 that is it will take more time. Similarly, for design database we now have optimistic estimate of 40, likely estimate of 45 days and pessimistic estimate of 60 days. So, we now have 40 days as optimistic estimate, 45 days as likely estimate and 60 days as pessimistic estimate.

Similarly, code database there we have 95 as optimistic, 105 as likely and 120 as pessimistic. The same thing we can do for all the other parts design GUI, code GUI, write user manual, integrate and test, everywhere we can now have three estimates rather than a single estimate. So, in activity network what we have seen is that we have only one estimate that is the likely estimate, we can consider it to be the likely estimate.

Whereas in the other cases, along with likely estimate we have optimistic and pessimistic estimates as well. Now, the advantage of having this PERT chart instead of a simple activity diagram is now here we can actually come up with better estimates of the project completion time along with some time overrun limits that we can estimate. So pessimistically what time it should take that is the longer time, optimistically what time it should take and the likely estimate that we have estimated earlier.

So, it will lead us to better estimation of the total time required to complete the project from this dependency diagram. So, PERT chart is a better way to represent the time schedule than activity networks. These are broadly the issues that are important for project managers when they try to estimate various things about a project. Just to recap, project management is a very important part of any project development.

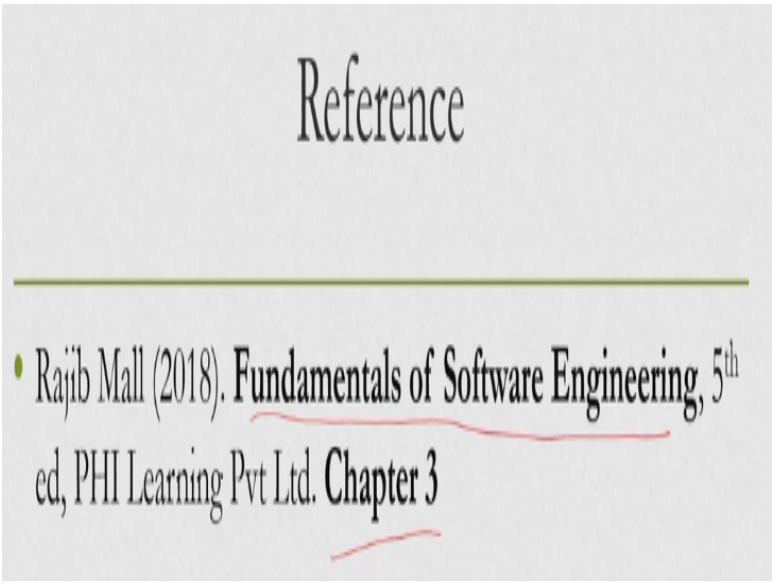
Earlier we have not discussed it, but it should be kept in mind that this is the first thing that should be done before the development activity starts because unless this stage is done properly, then the allocation of resources and estimation of requirements cannot be done properly. So, as a project manager, a project manager is primarily required to estimate project size, cost, duration, effort.

Then prepare schedules and allocate resources for smoothly carrying out the development process. Now, in this endeavour manager can take help of several graphical notations to plan and represent those plans. For example, they can make use of the PERT chart or Gantt chart

or both together or activity diagram and task hierarchy, etc. For estimation of cost, time, etc., COCOMO model can be used.

It may be noted here that whatever concepts we have covered in this lecture are covered in very brief manner to keep the discussion simple as this is not the primary focus of our course. The primary focus of our course is the development process for interactive systems rather than how to manage the development process.

(Refer Slide Time: 46:23)



So, for more details, you may refer to this reference Fundamentals of Software Engineering chapter 3, there you will find these concepts in more details plus you may refer to any other book on project management for managing complex software projects. I hope the material that I covered in this lecture is of interest to you and you have understood the concepts.

Of course, we kept it at a very introductory level and for more details you have to refer to the references, other resources. But I hope you enjoyed the material and I look forward to meet you all in the next lecture. Thank you and goodbye.