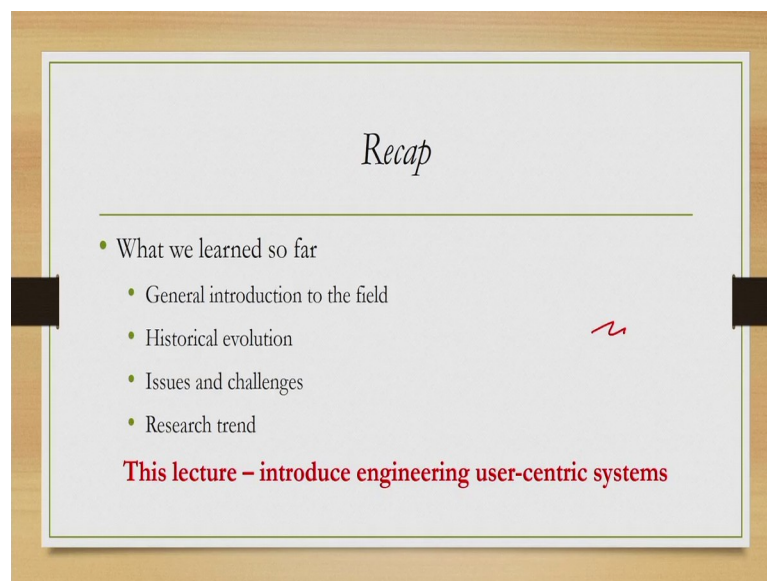


User Centric Computing for Human-Computer Interaction
Prof. Samit Bhattacharya
Department of Computer Science & Engineering
Indian Institute of Technology, Guwahati

Lecture - 04
User-Centric Design and Software Engineering

Hello and welcome to the 4th lecture for the course user Centric Computing for Human Computer Interaction.

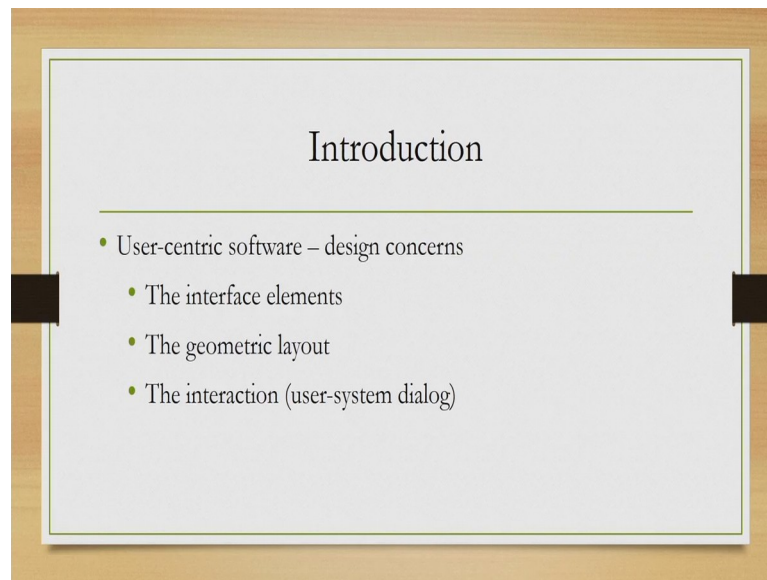
(Refer Slide Time 00:47)



So, before we proceed further, let us briefly recap what we have learned so far. So, in the earlier lectures we mostly covered the introductory concepts; namely what the field is all about, what are the issues and challenges, brief historical evolution and the research trends in this area.

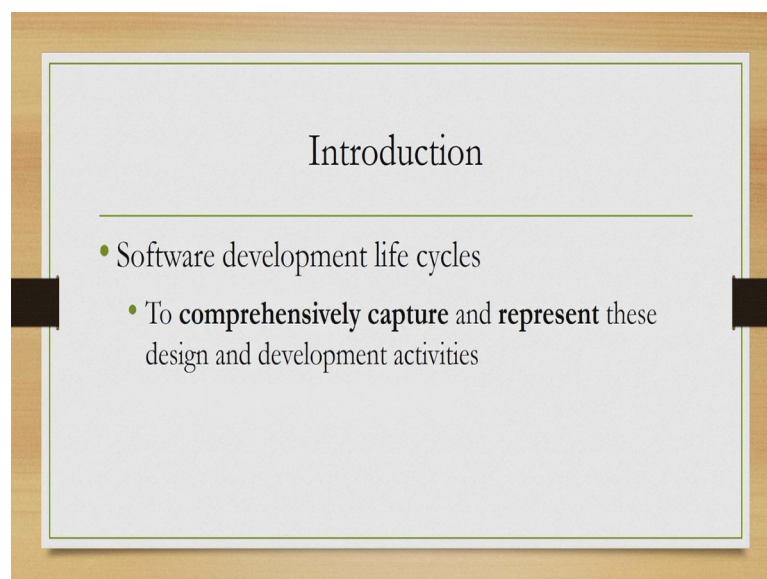
Now, today we will introduce the one of the major content that we are going to cover in this course namely; the user centric design through the perspective of a software engineer. So, how to design user centric systems with an engineering perspective?

(Refer Slide Time 01:31)



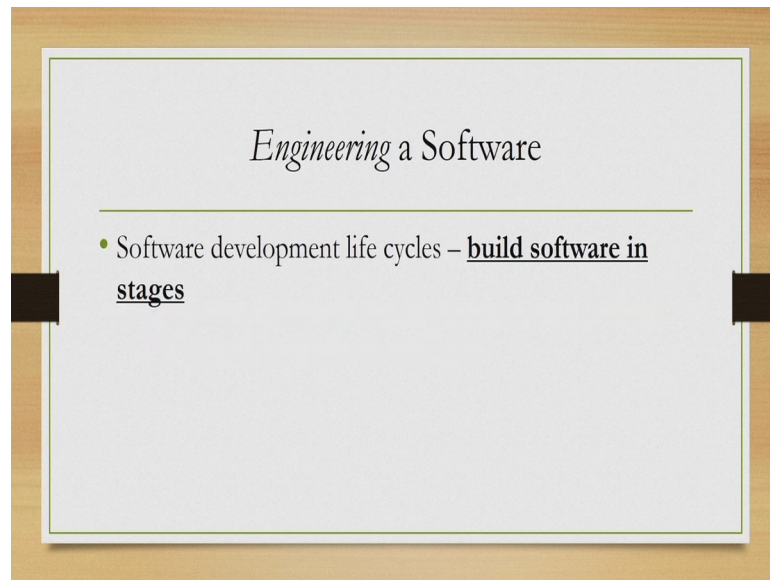
Now, let us start with the design concerns; what we are trying to design in user centric systems. So, there are three major things; one is the interface elements, one is the geometric layout, how the elements are arranged on the interface and the third is the interaction how to access the elements, this is also known as user system dialog. Now, these three major activities, each of these in itself is a major activity. Now, how to manage these three activities we need some way to visualize the management.

(Refer Slide Time 02:17)



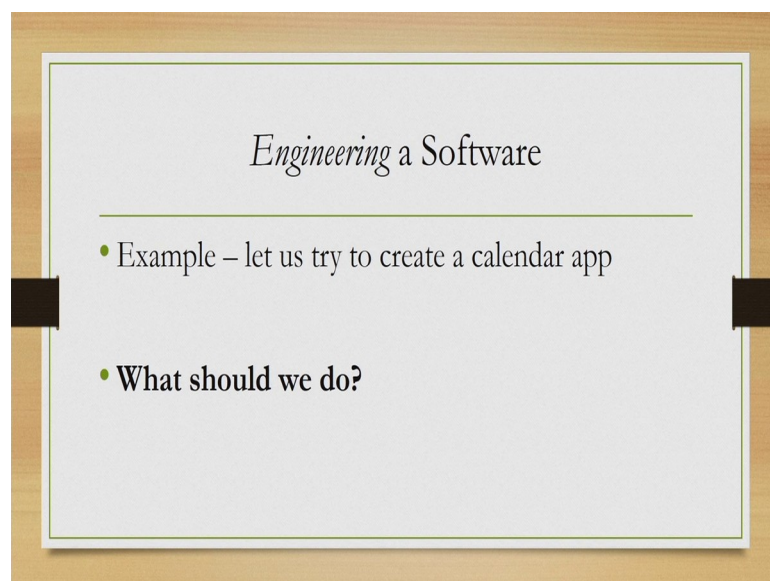
And one good way to visualize that is, to propose a development lifecycle. So, that software development lifecycle can give us a good representation of how to manage the three design activities. So, it helps us to comprehensibly capture and represent these activities.

(Refer Slide Time 02:40)



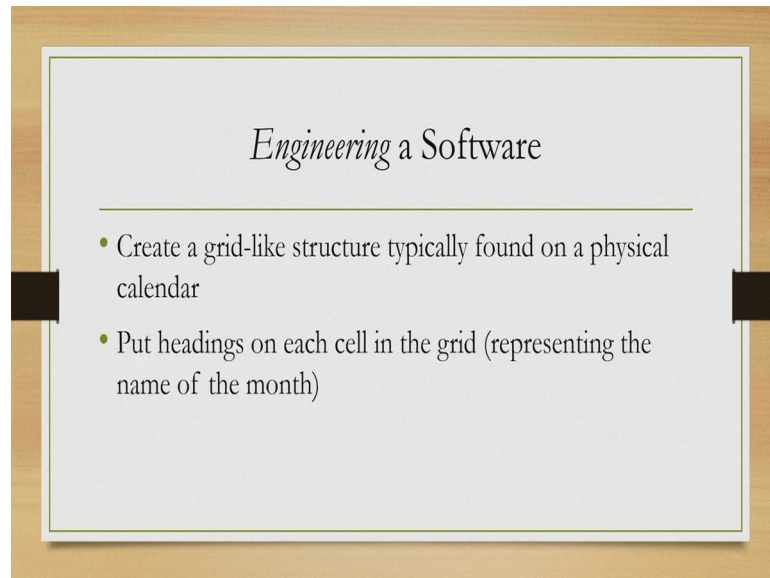
Now, what is the software development lifecycle? It is a way to represent the development of a software in different stages.

(Refer Slide Time 02:53)



Now there can be many such life cycles. But before we go into the details of the life cycle, let us try to understand the difficulty in developing a software. Let us try to create a calendar application. What should we do to develop the calendar?

(Refer Slide Time 03:14)

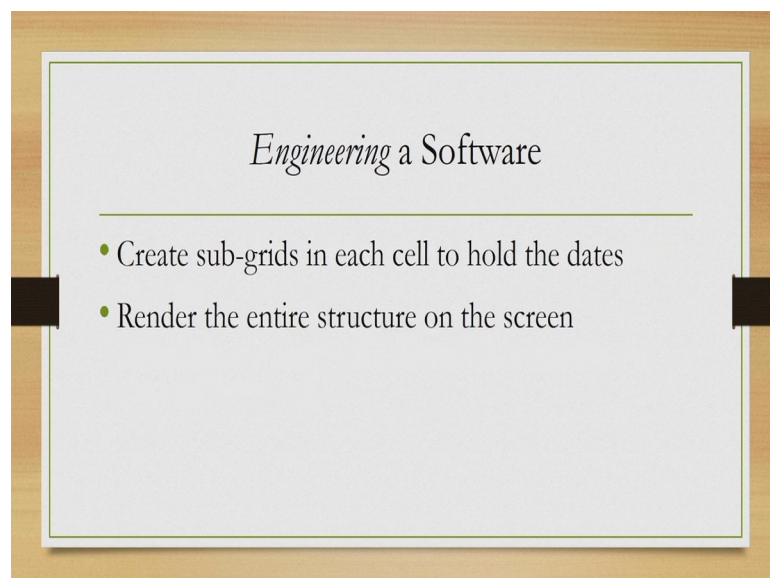


Engineering a Software

- Create a grid-like structure typically found on a physical calendar
- Put headings on each cell in the grid (representing the name of the month)

Now, if we want to develop a calendar, the way we see a calendar; then typically we should try to create a grid like structure, because calendars typically comes in the form of a grid. Then we should put headings on each grid element or grid cell that were probably represent the name of the month.

(Refer Slide Time 03:41)

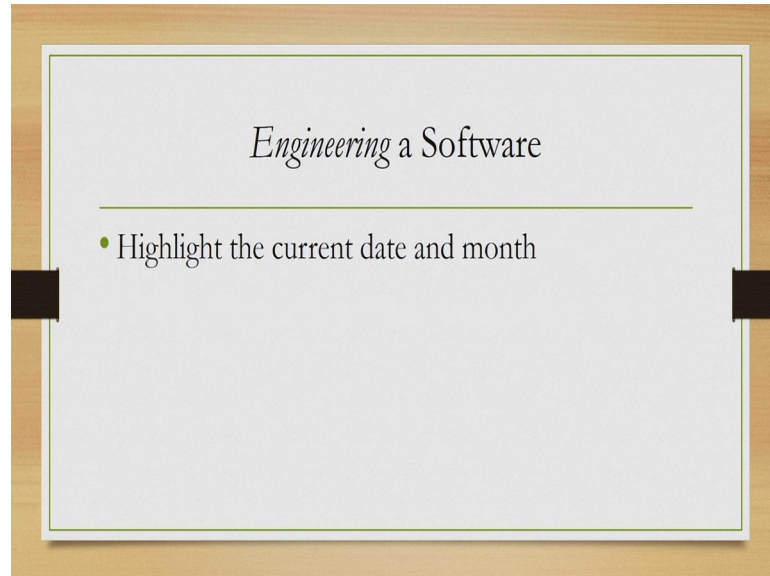


Engineering a Software

- Create sub-grids in each cell to hold the dates
- Render the entire structure on the screen

Then create sub grids within each cell, probably to represent the days of the month or the dates and render the entire structure on the screen.

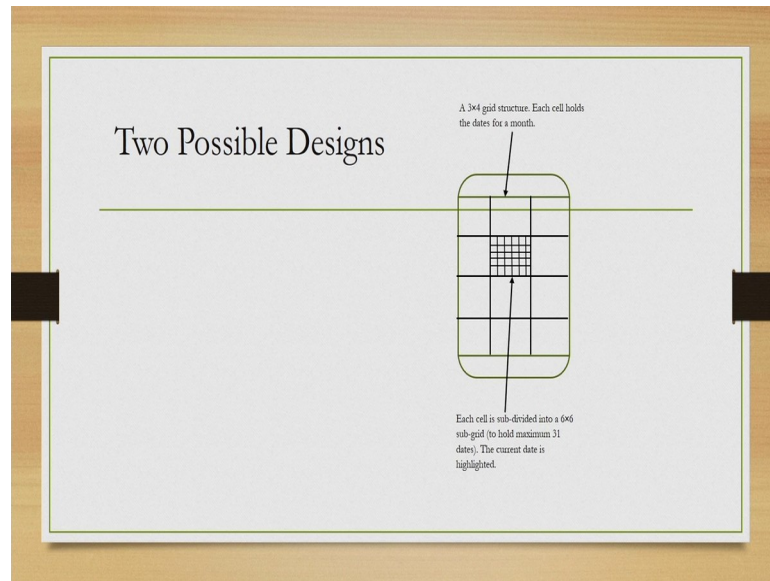
(Refer Slide Time 03:57)



And additionally to make the life of the user simpler, we may also highlight the current date and month. So, that is what we want. Now, suppose you are asked to do it on a mobile screen, on a small screen; what you will do? There are actually different ways to do it. The challenge is to accommodate all the dates of the month on the same screen. So, we can probably come up with different design options.

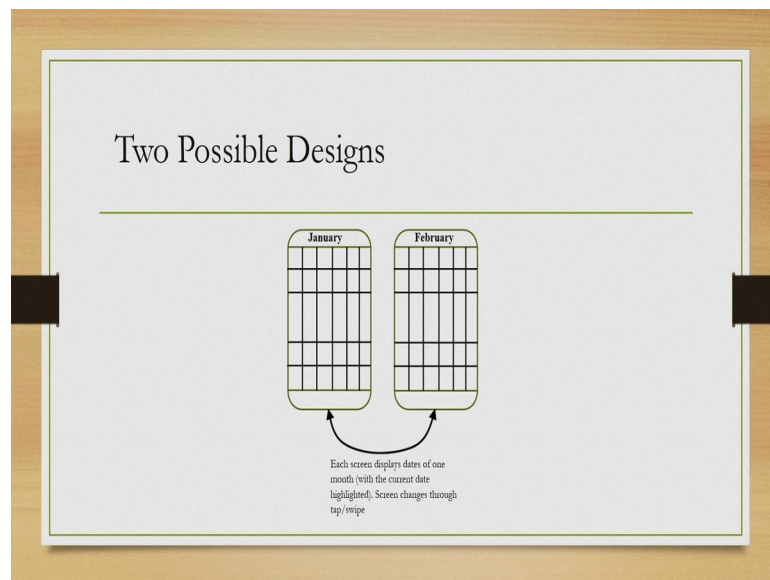
Let us discuss two of those; in the first option, we may try to have a structure like this where we have the grid elements and the sub grid elements shown on the same screen. But of course, the sub grid elements will be very small.

(Refer Slide Time 04:30)



So, the upper level grid will have 12 cells representing 12 months and inside it grid there will be cells to hold maximum 31 dates and minimum 28 dates and the month; and the date or the current date we can highlight in this display.

(Refer Slide Time 05:24)



There can be a second possibility that instead of showing everything on the same screen, we divide it up. So, in each screen we will show the days of the month. And so, at the upper level we will have 12 cells; suppose we touch on a cell and the days appear, the

month disappear and only the days are shown for that particular month which we touched.

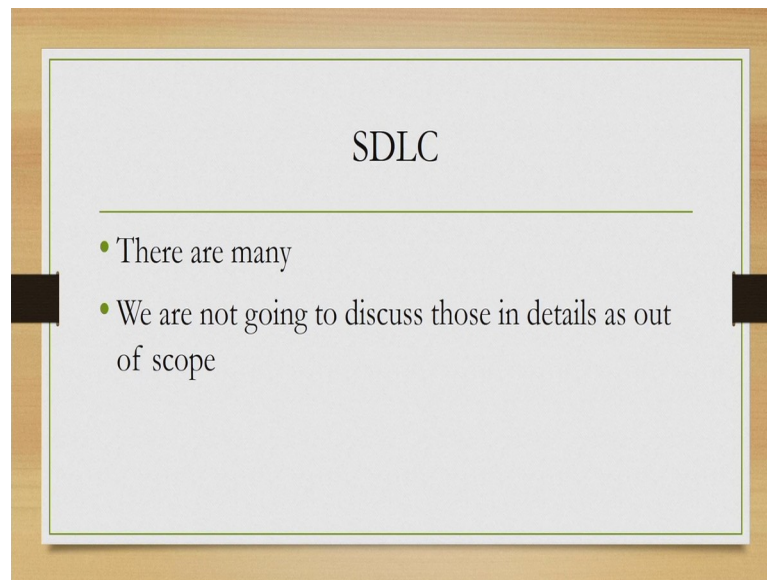
So, there is a screen change from month view to date view and then from date view to month view again through the presence of some back option. This is another design; which design should we choose and how should we choose that design. We should not arbitrarily decide based on our likes and dislikes; our means, the designers. We should follow certain stages to choose a design that is likely to be acceptable to the users.

(Refer Slide Time 06:31)



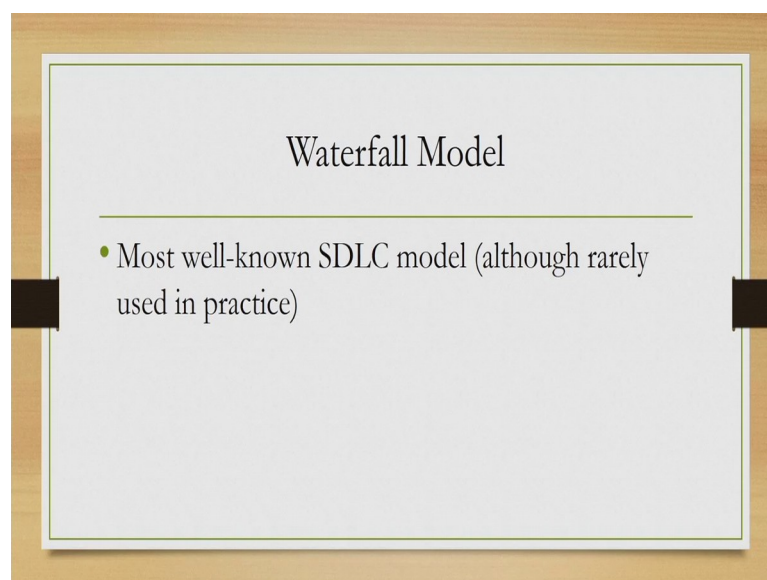
Now, when we want to choose a design, the challenge is what should guide us in our choice. So, that is a complex question with complex answer. And we require some systematic approach to choose a particular design and there the software development life cycles help.

(Refer Slide Time 06:59)



So, these life cycles tell us; what are the stages we should follow to come up with a final design. Now, there are many software development life cycles. And probably if you have done software engineering course in your undergraduate days or as part of your undergraduate program, then you are probably aware of many such development life cycles. We are not going to discuss all those here, that is out of scope of this lecture.

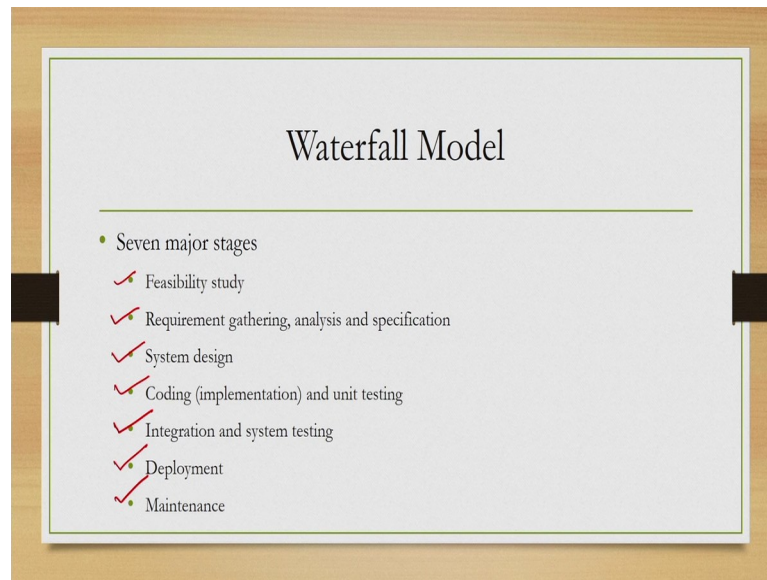
(Refer Slide Time 07:40)



Instead to demonstrate the idea of SDLC, Software Development Lifecycle will refer to the pedagogically most popular development lifecycle that is known as the waterfall

model. Of course, as probably most of you know that this model is not used in practice, it is only used for pedagogical purposes. It used to be used earlier, but then lots of problems were found and different models came. But here we will stick to this model to highlight the issues with this traditional way of software development.

(Refer Slide Time 08:04)



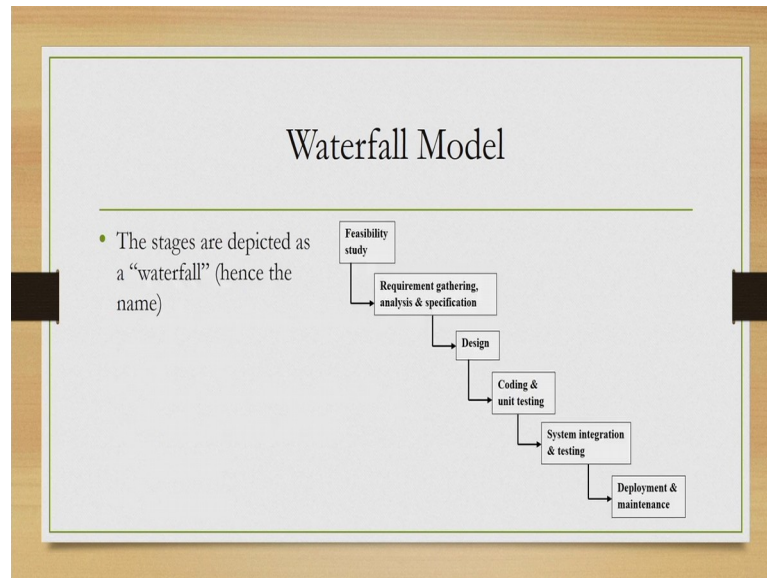
Now, I think most of you may be aware of this; if not, let us just recap what this model is. In this model, software development is viewed as a combination of seven stages. The seven major stages are; first stage feasibility study; then we have requirement gathering, analysis and specification; then we have design of the system, then we have coding or implementation unit testing; then comes integration and system testing; finally, deployment and the last stage is maintenance.

So, in the feasibility study we try to find out the specs or the feasibility of carrying out the project; and requirement gathering we try to find out by talking to end user or taking feedback from end users and our own analysis; what are the things that we need to develop in the software; then that requirement is translated to a design of the system, typically in the form of data flow diagram DFD or UML class diagrams.

Once the design is ready, we go for coding. So, the design actually tells us the modules and sub modules. So, we go for coding at unit level test it and then we integrate those units to develop the modules and the whole system and then go for testing at module

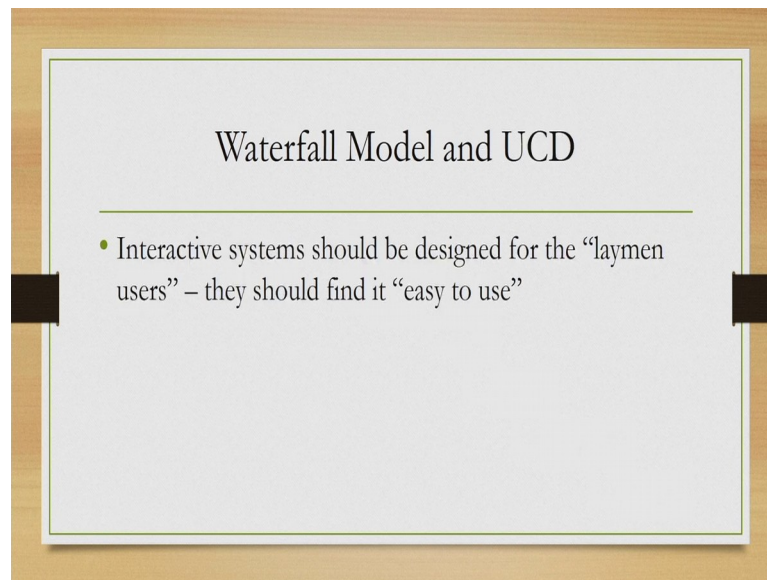
level or whole system level. Once those are done ready tested, we go for deployment and the last stage is maintenance.

(Refer Slide Time 09:47)



So, these stages are shown when they are depicted, they are depicted in the form of a waterfall; like the way we have shown here. See feasibility study followed by requirement gathering, followed by design, followed by coding and unit testing, followed by system integration and followed by deployment and maintenance. This appearance gives the model its name, this particular development lifecycle model is named waterfall; although we can, it is not necessary mandatory to show it in this form, but that is the way traditionally it is shown.

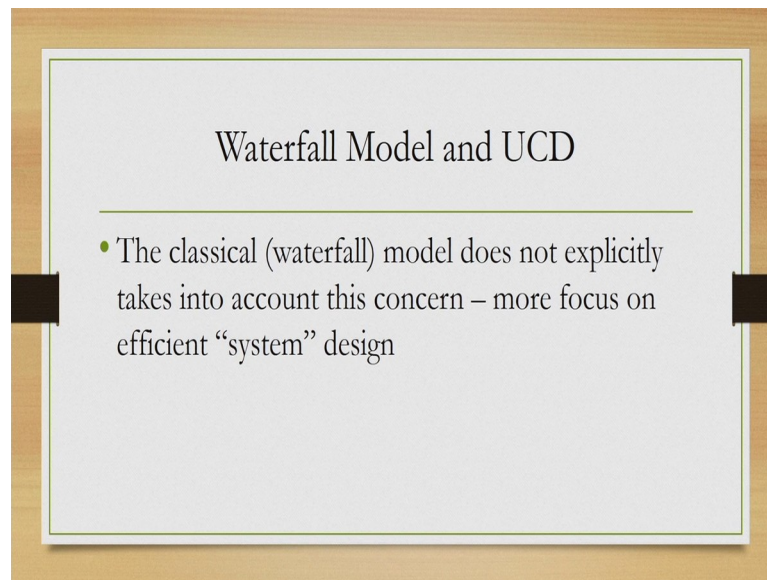
(Refer Slide Time 10:16)



So, if we try to develop a user centric product or user centric software, systems software through the user centric design approach; will there be any problem if apply the waterfall model. Before trying to answer that question, let us try to understand what we want to do in UCD or user centric design.

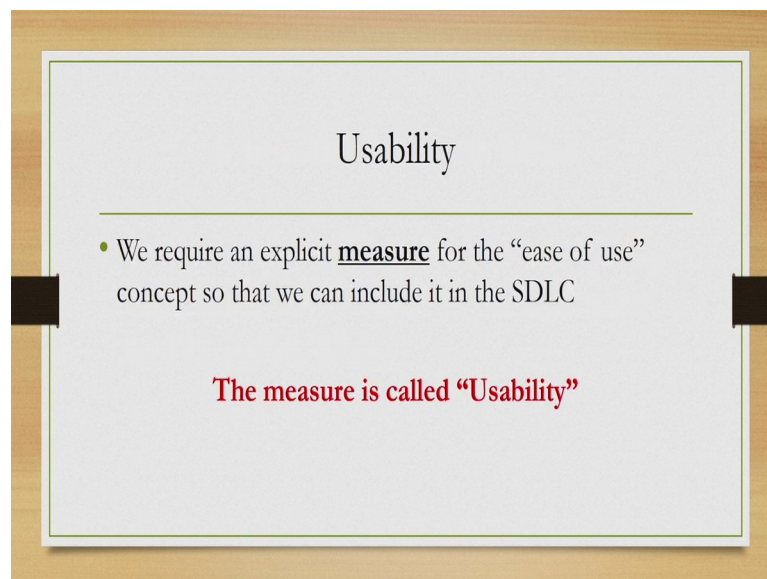
So, essentially our objective is to design for the laymen users; the users who do not have the technical knowhow for the particular system they are using, but they want to find it easy to use. So, that is our objective design a system, so that laymen users find it easy to use. Now, the classical waterfall model does not explicitly incorporate the concerns or characteristics of laymen users.

(Refer Slide Time 11:04)



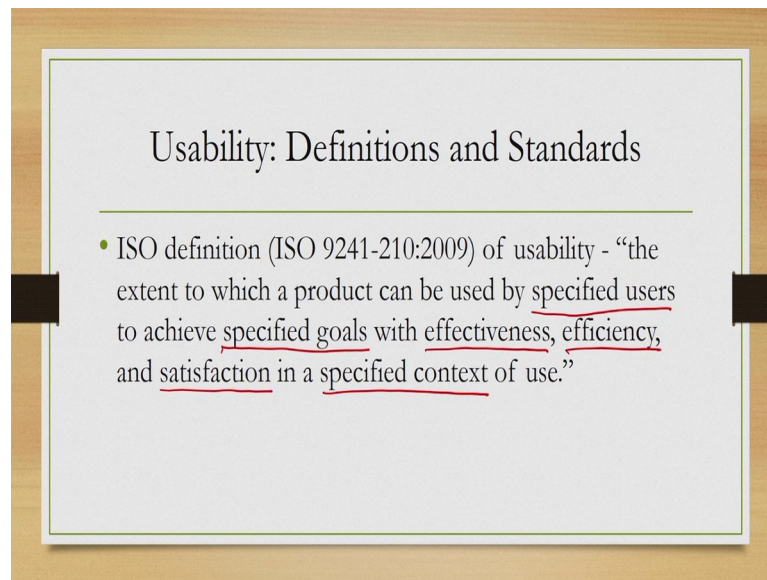
Here the focus is more on efficient system design.

(Refer Slide Time 11:16)



Now, in order to incorporate the concerns or the characteristics of laymen users in a design development model, like the waterfall model, we need somehow to measure the user characteristic; Otherwise it is very difficult to incorporate this concept in a software development lifecycle. So, there is one measure of this ease of use which is known as usability. Let us first try to understand the concept of usability, then we will come back to the idea of UCD and its relation to waterfall model.

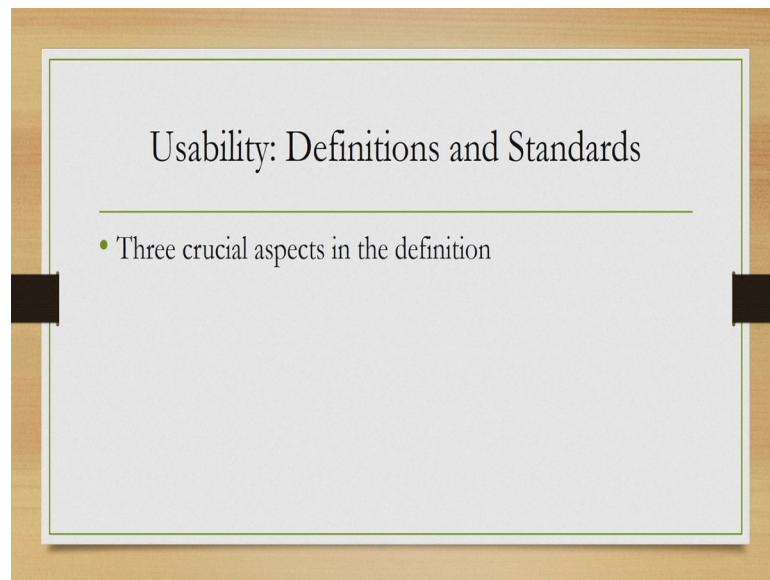
(Refer Slide Time 11:57)



So, what is usability? So, so far we are referring to the terms intuitive, easy to use, acceptable, etcetera to refer to products that the end users find again quote unquote easy to use. Now, these are (Refer Slide Time: 12:14) terms; instead if we have a unambiguous definition of this easy to use, then it will be helpful, usability gives us such a definition. Now usability is a concept which has an ISO or international standards organization definition; the standard number is 9241 210 2009.

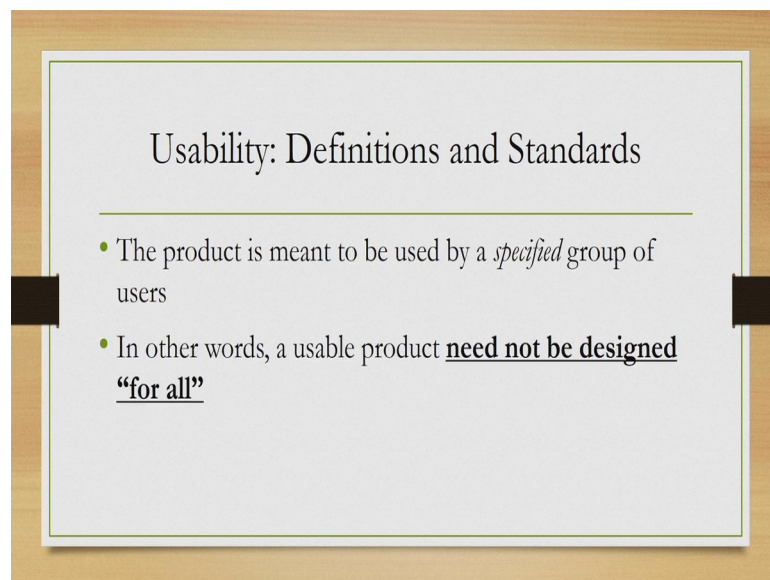
According to this definition; usability is a concept or a measure which measures the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use. Mark the use of the phrases specified users, specified goals, and specified context; along with these words effectiveness, efficiency and satisfaction.

(Refer Slide Time 13:28)



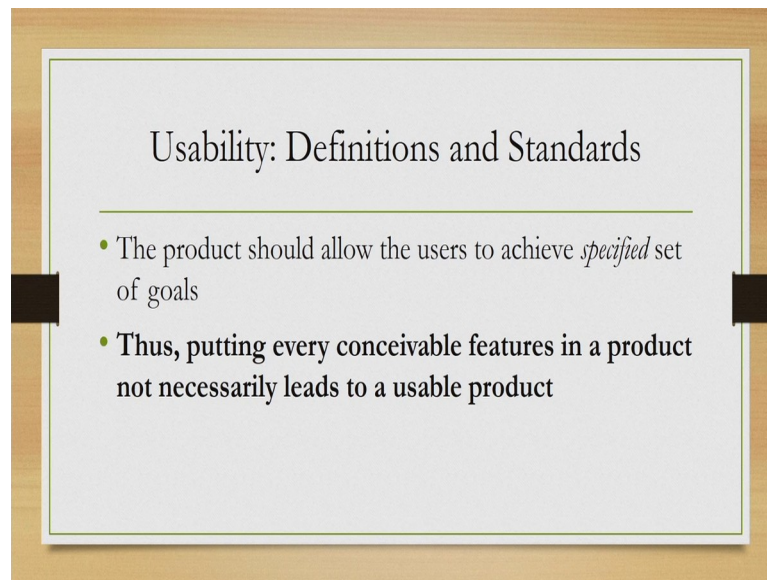
So, what these indicate? Let us see one by one. So, the definition has three crucial aspects.

(Refer Slide Time 13:36)



First of all, the product is meant to be used by specified group of users. So, when I say specified group of users; I implicitly what I mean is that, a usable product need not be designed for all this is very important. So, when you are asked to design a user centric product system or a software, you should not assume that you can design it for everybody in this world. So, it should always have a specified group of users.

(Refer Slide Time 14:13)

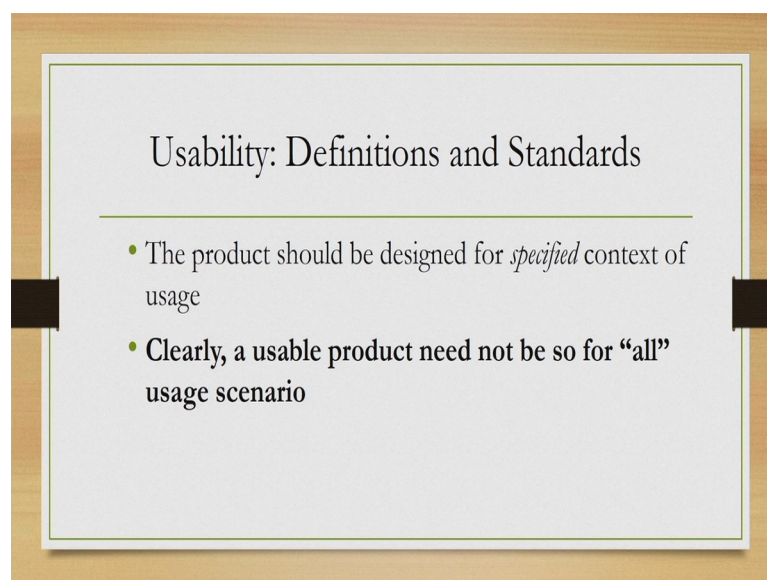


Usability: Definitions and Standards

- The product should allow the users to achieve *specified* set of goals
- **Thus, putting every conceivable features in a product not necessarily leads to a usable product**

The second aspect is; that the product should allow the users to achieve specified set of goals. So, again when you are asked to design a user centric software or a product; you should not try to incorporate as many features as possible based on your decisions. You should find out what the user tries to achieve and accordingly you should try to provide support for those activities only, should not include everything.

(Refer Slide Time 14:56)



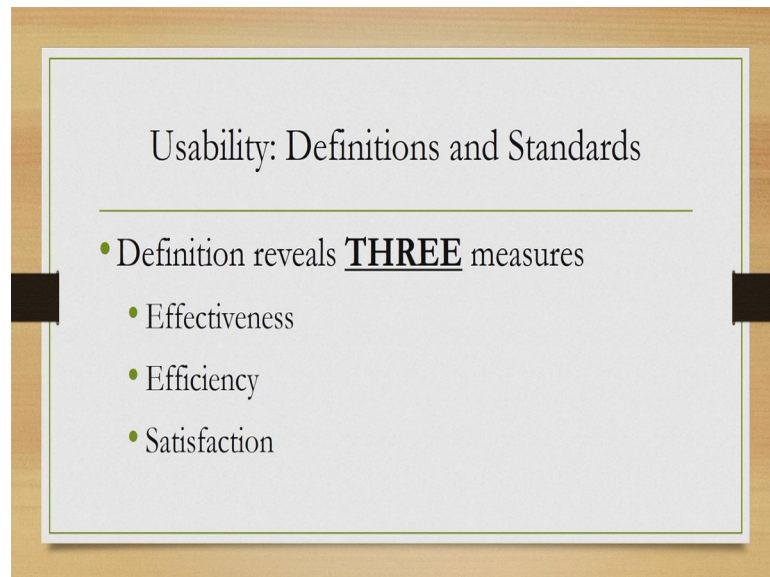
Usability: Definitions and Standards

- The product should be designed for *specified* context of usage
- **Clearly, a usable product need not be so for “all” usage scenario**

And thirdly the products should be design for specified context of use. So, this is a very interesting thing. So, it implies that your product whatever you develop, whatever

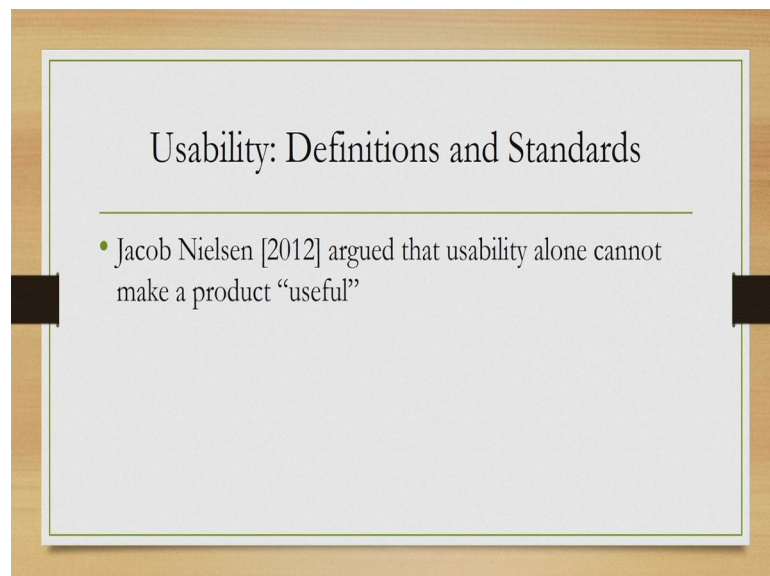
software you develop need not be usable in all the context of you. So, it may be usable in one context, may not be usable in another context. So, if you are planning to design a product that is usable for all usage scenarios, then you may never be able to achieve your objective.

(Refer Slide Time 15:31)



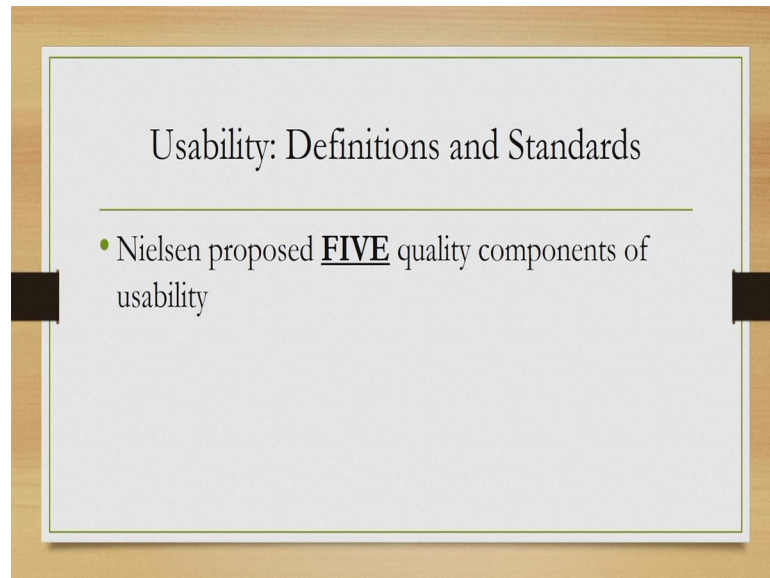
Along with these three crucial aspects that ISO definition of usability provides for three measures, specific measures these are; effectiveness, efficiency and satisfaction. However, the definition does not say how to measure these three quantities.

(Refer Slide Time 15:55)



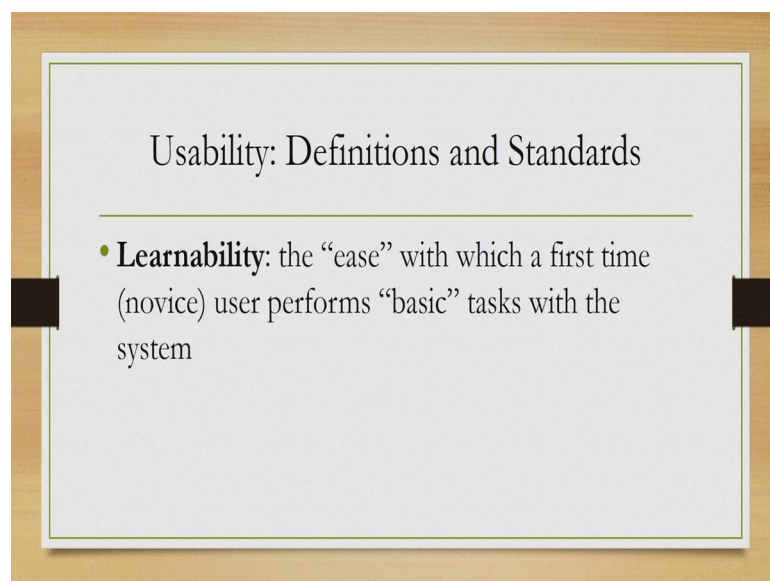
Now, in 2012 Jacob Nielsen argued that only usability is not suitable to make a product useful. So, when we are using the term useful whether a product is useful or not. So, we may not be able to say anything about that based on only the concept of usability. So, instead what he proposed is that, any acceptable or useful product should have two quality components attributes; one is usability and the other is utility.

(Refer Slide Time 16:37)



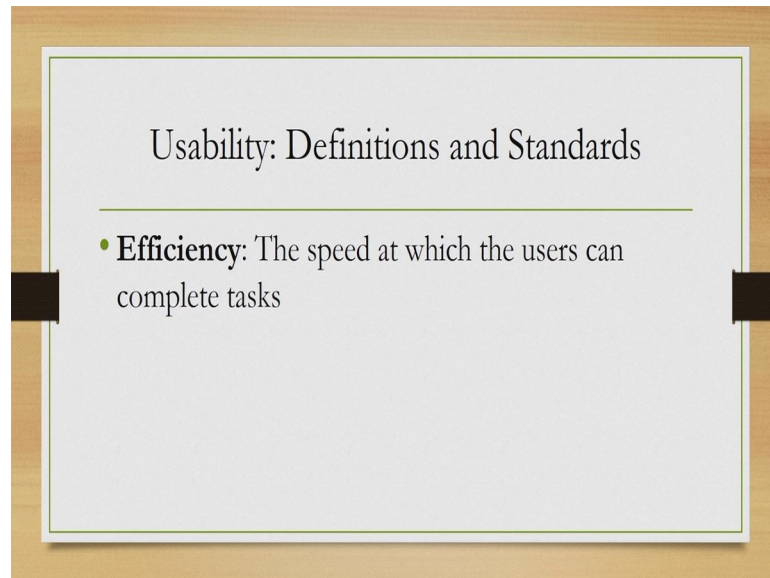
Now, according to Nielsen we can actually have five quality components for usability.

(Refer Slide Time 16:48)



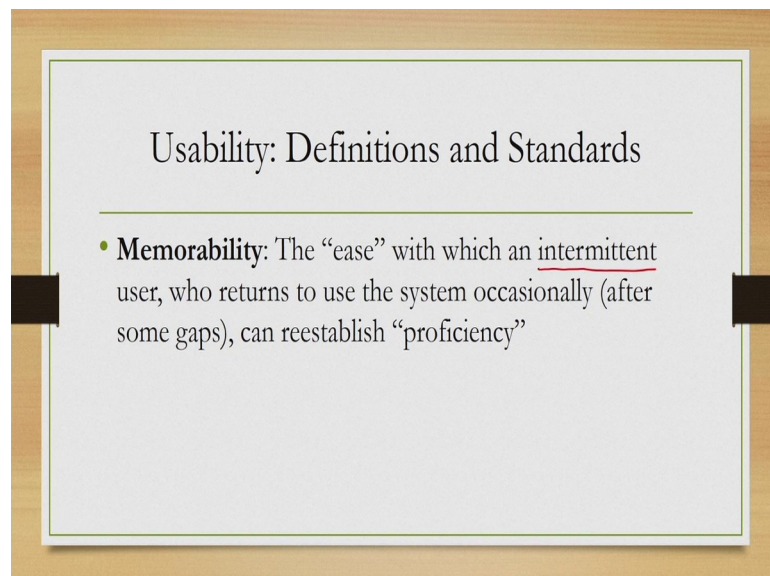
One is learnability; it is a measure of the ease with which a first time user or a novice user performs basic tasks with the system. So, recollect from our discussion in earlier lectures that we can divide users into three groups; novice, intermediate and expert. So, here learnability actually is a measure of the ease with which a first time or novice user can perform basic tasks.

(Refer Slide Time 17:21)



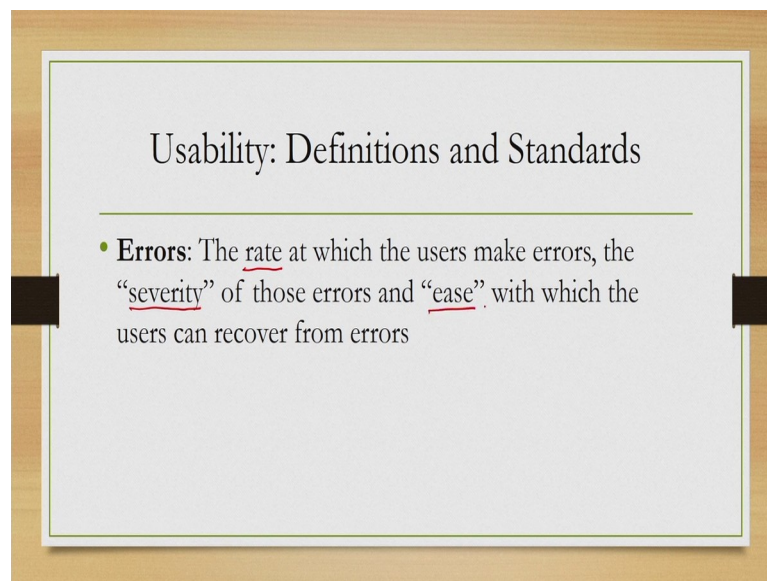
Next is the measure; next is the quality attribute efficiency which is a speed, which is the speed at which the users can perform or complete the tasks.

(Refer Slide Time 17:34)



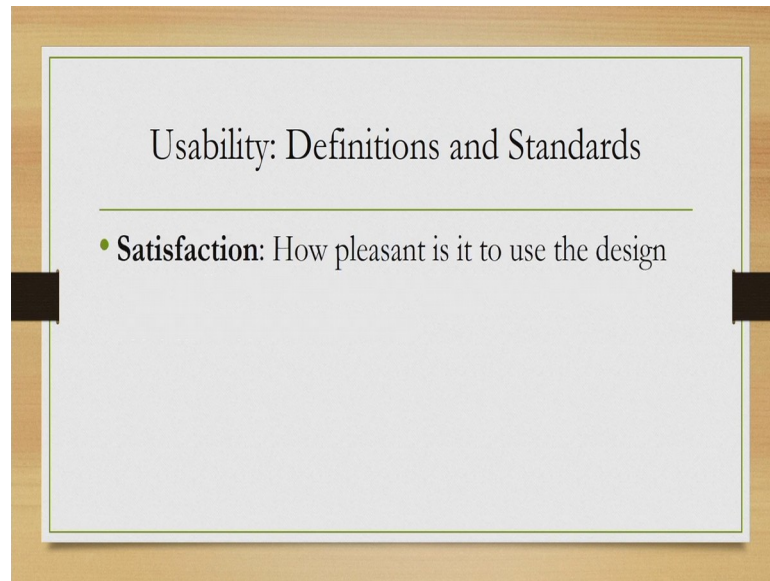
Then comes memorability; it is another measure which measures the ease with which an intermittent user, among the three groups of users an intermittent user who returns to use the system occasionally after some gaps can reestablish proficiency. So, note the emphasis given on intermittent users. So, memorability measures the ability of a user to recollect what he or she has learned in the previous uses and utilize that knowledge effectively, even if the user is not using it frequently.

(Refer Slide Time 18:16)



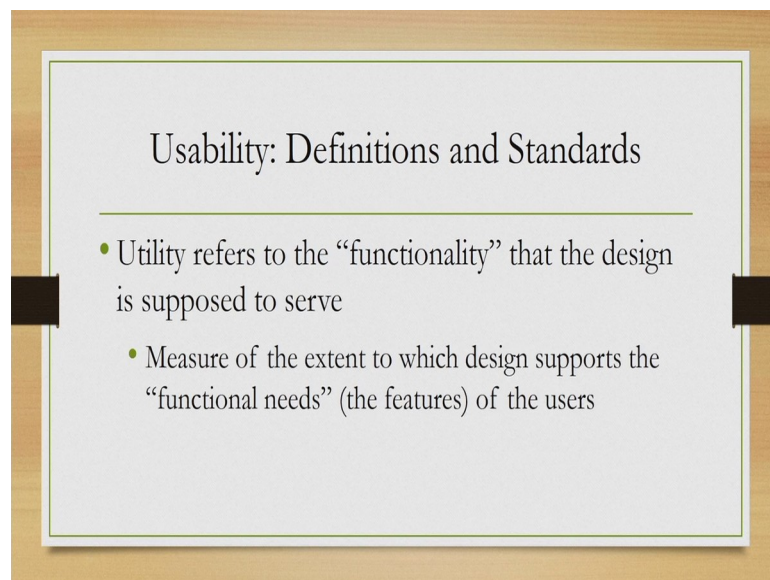
Then comes errors; the rate at which the users make errors, the severity of those errors and the ease with which the users can recover from the errors. All three are important; the rate, the severity of the errors and the ease with which a user can recover from an erroneous situation.

(Refer Slide Time 18:43)



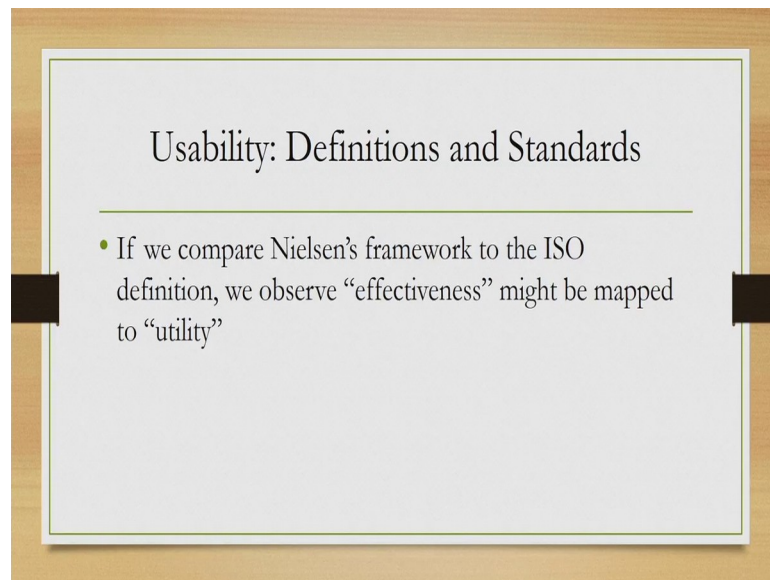
And finally satisfaction; it is a measure of the pleasantness, how pleasant the interface is or how pleasant it is as perceived by the user.

(Refer Slide Time 19:02)



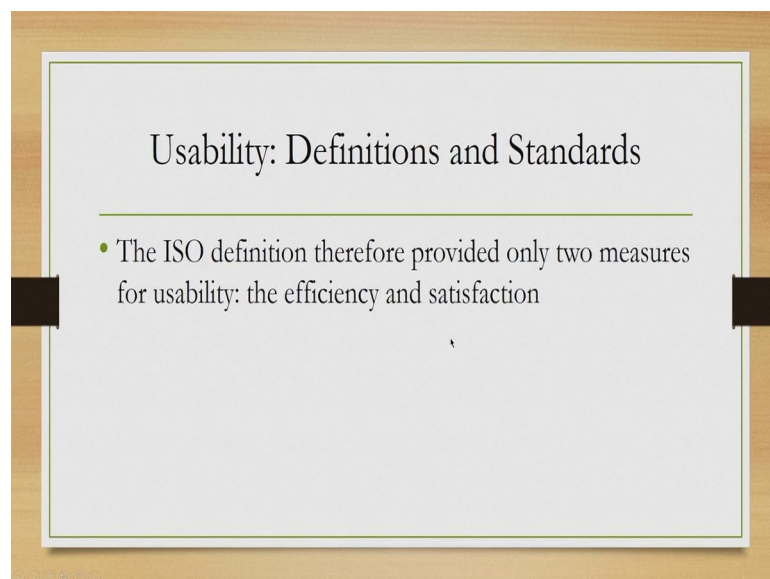
So, along with this five quality attributes, there is another measure proposed by Nielsen as I said; that is utility which refers to the functionality that, the design is supposed to serve the extent to which the design supports the functional needs or the features.

(Refer Slide Time 19:22)



Now, if we compare Nielsen's definition or Nielsen's framework to the ISO definition, then we can observe that effectiveness is map to utility. The ISO definition content the notion of effectiveness which we can roughly map to the utility proposed by Nielsen.

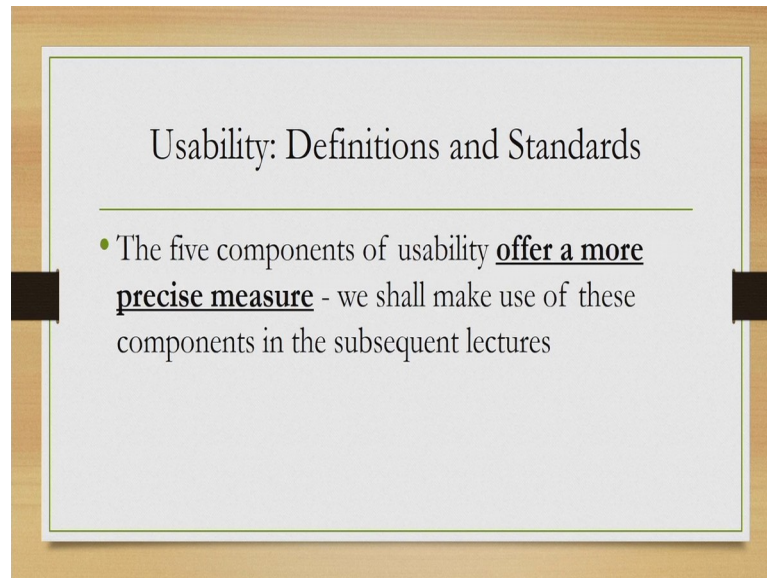
(Refer Slide Time 19:42)



And the other two entities in the definition namely; efficiency and satisfaction can be mapped to usability.

So, according to the ISO definition we can have two measures of usability which is efficiency and satisfaction. But if we go by the Nielsen's definition, then we can have five measures; memorability, learnability, error rate efficiency and satisfaction.

(Refer Slide Time 20:06)

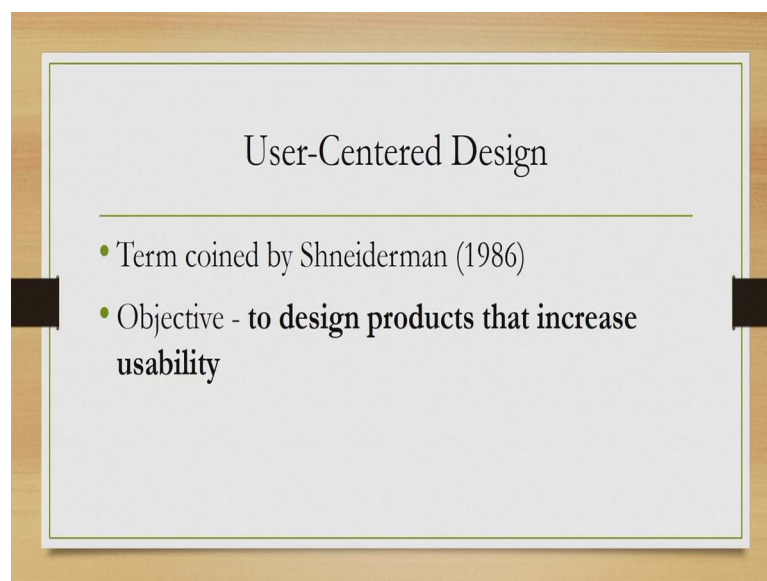


Usability: Definitions and Standards

- The five components of usability **offer a more precise measure** - we shall make use of these components in the subsequent lectures

Since this gives a more precise definition of usability, it is easier to deal with this Nielsen's definition than the definition of ISO. So, will. So, in our subsequent lectures we will stick to this definition of usability and these five components of usability.

(Refer Slide Time 20:38)

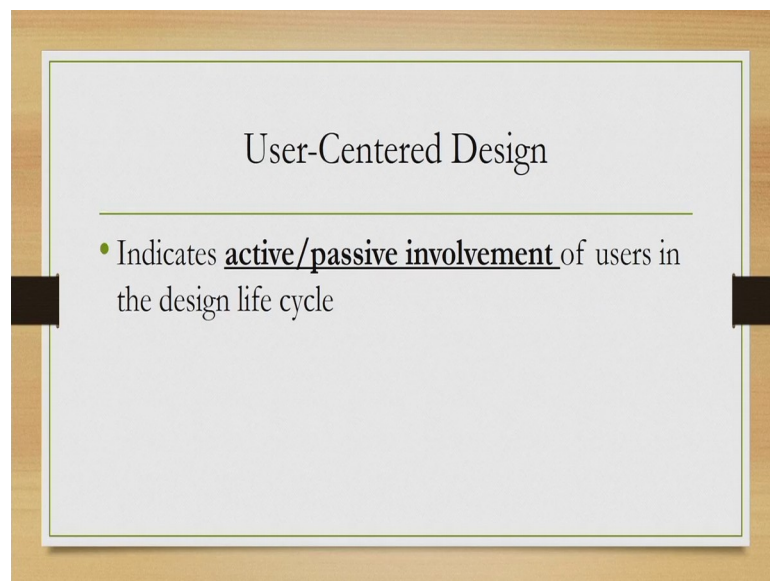


User-Centered Design

- Term coined by Shneiderman (1986)
- Objective - **to design products that increase usability**

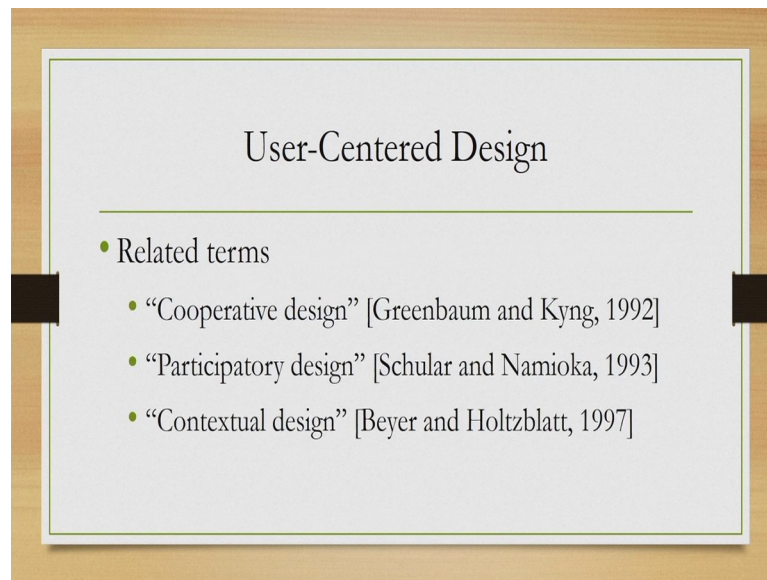
Now, once we know how to measure usability in terms of these five components; the next is the relation of usability to user centric design. So, as we have mentioned in one of the earlier lectures, this term user centric design was coined by Shneiderman in 1986; and the objective of user centric design is, to design products that increase usability. So, our objective would be to have a product or have a software that, increase the usability of the interface of the software.

(Refer Slide Time 21:15)



Now, user centric design, a typically indicates that the user, end user is either actively or passively involved in the whole design process.

(Refer Slide Time 21:28)



So, that is the crucial considerations in user centric design that you involve somehow the users; either as active member of the design group or as passive member to get information feedback on the usability of the design.

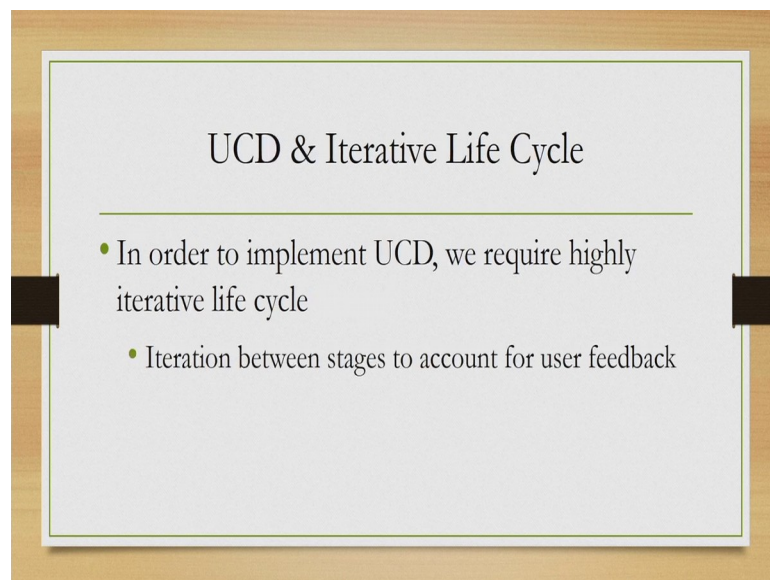
Now, this user centric design and this concept; there are many related terms to this concept, like cooperative design proposed by Greenbaum and Kyng in 1992; then participatory design proposed by Schular and Namioka 93; and contextual design proposed by Beyer and Holtzblatt in 97. All these terms more or less refer to similar idea that, you involve the user in the design process.

(Refer Slide Time 22:27)



ISO in its standards preferred to use the term human centered design rather than user centered design.

(Refer Slide Time 22:36)



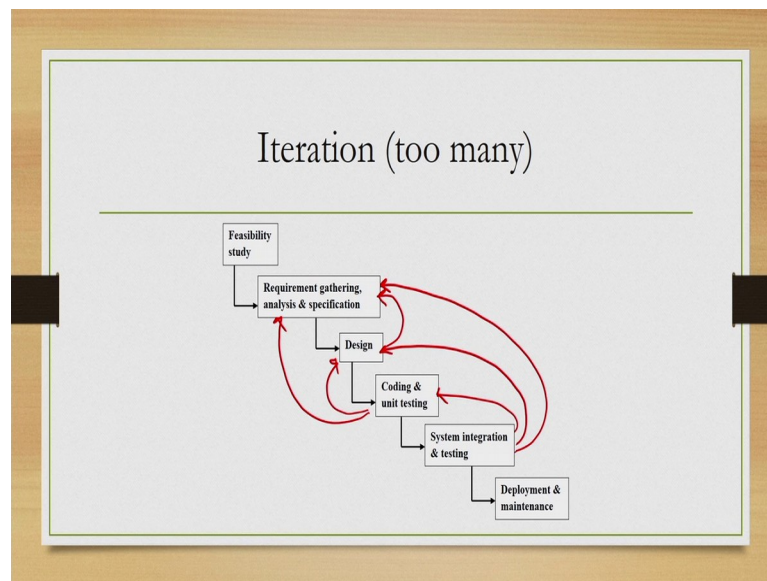
Now, let us come back to where we started; that we want to manage or visualize the design in terms of software development lifecycle. Now, this lifecycle we have seen the waterfall model has one of the lifecycle models. And now we know the concept of usability and how usability is relative to user centered design; where the main concern is

that you involve the user in the design process either as a active member or as a passive member.

So, now let us see how that is possible in the context of waterfall model. In the waterfall model whether can we incorporate the user to increase the usability of the end product? Now, in order to implement user centric that process, we require highly iterative lifecycle. So, because at every step we need to get some feedback; whether what we are doing is making the product usable or not.

Now, iteration between stages are required in any lifecycle whichever we choose to implement user centric design process, so that we can get user feedback and make a usable product.

(Refer Slide Time 23:54)

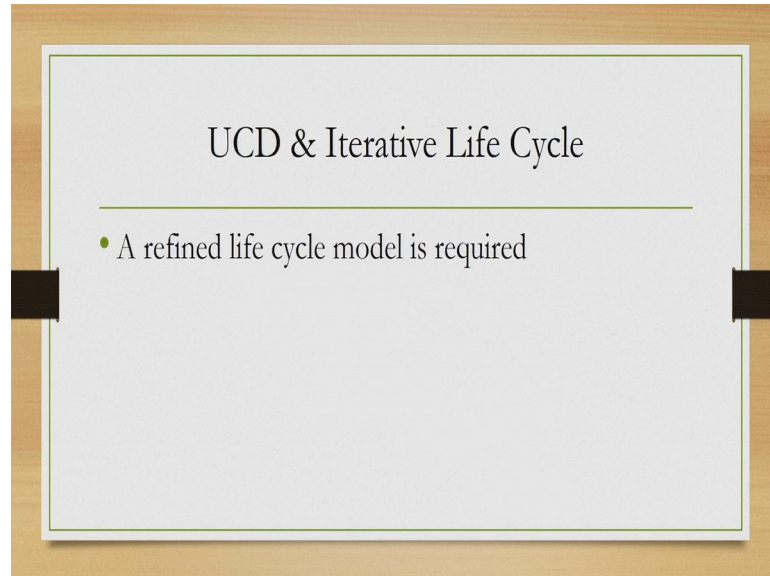


So, if we relook at the waterfall model, then to get user feedback we need lots of going back and forth between the stages. Say for example, requirement gathered for a usable product and we designed; but then we found certain problem and we may need to go back to the requirements stage. After design we came to coding, we may need to go back to design or we may need to go back to requirement gathering stage.

In fact, after system integration and testing we may need to go back to design, we may need to go back to requirements stage or even we may need to go back to coding stage. And that actually happens even after deployment as well. So, we may need to refine

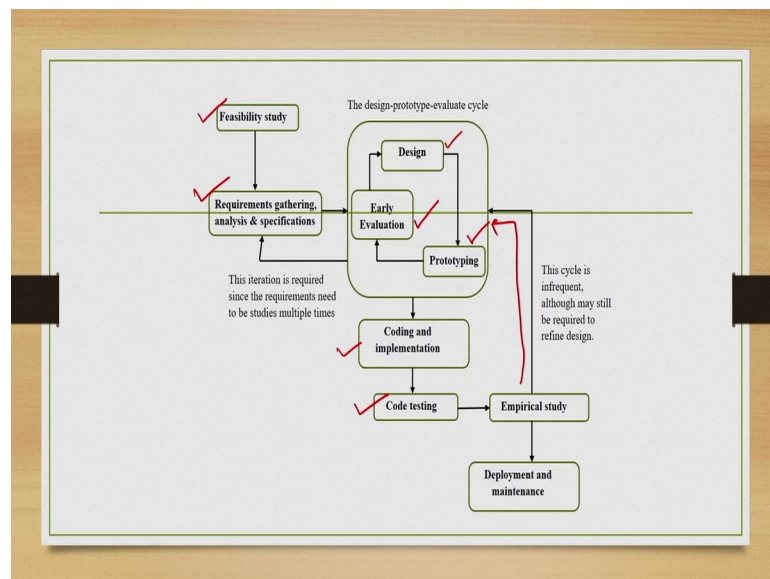
certain things, so that the product life increases. So, as you can see there can be lot of back and forth movement between the stages.

(Refer Slide Time 24:59)



So, pure waterfall model may not be suitable. In fact, pure waterfall model is not suitable to implement an UCD process and we require a refined lifecycle model for software development.

(Refer Slide Time 25:11)



And one such model we can propose in this way. So, as you can see, we start with the feasibility study as in waterfall model; then from there we come to the requirement

gathering, analysis and specification stage, again as in the waterfall model. But from there we go to a set of sub stages which are iterative.

So, within this set there are three sub stages; we design, then we prototype, create a prototype of the design, then go for early evaluation of the design; based on that we refine the design, then again prototype, then perform early evaluation. And this cycle goes on till we get satisfactory result in the early evolution, where which indicates that no further modification may be required.

Now, this is a iterative cycle. So, after we come out of this cycle we go for coding an implementation followed by code testing. Now, after that again a new stage, that is empirical study. So, this study involves users. So, from users we directly want to get feedback. So, involve them in the study that is known as empirical study and after that we go for deployment and maintenance.

Now, in the empirical step study we may find some problems that is quite likely. So, we may need to go back to this iterative cycle again, where we refine the design, then prototype, then go for early evolution and the cycle is repeated till we satisfied to some extended come out of the cycle. So, with the typical waterfall model we have difference at many stages; like this the cycle of many sub stages, then from empirical study this going back to the cycle.

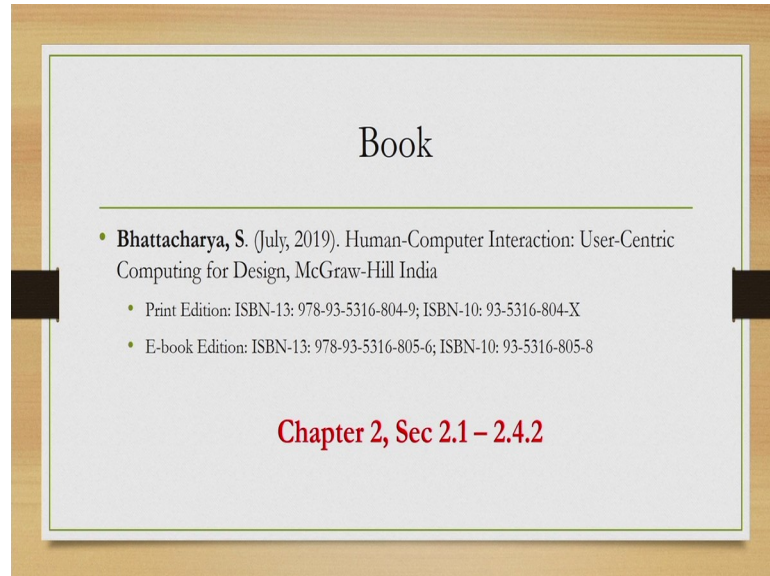
In the requirement gathering the way it is typically done for non-user centric software's; there is a lot of change in the user centric software where we perform requirement gathering, analysis in a different way, in a very prominently or markedly different ways which we will discuss in subsequent lectures. So, there are a lot of changes.

And overall this type of lifecycle model gives us an idea of how we can implement user centric design process to develop a product or a software, in our case a software; that actually is or that will succeed in increasing the usability of the interface or the interactive system. So, that is a basic idea of what we meant by an engineering approach to user centric system design.

So, here our engineering approach consists of this iterative lifecycle which is very essential to implement user centric software to increase the usability of the software. In

subsequent lectures we will see how these different stages are performed, what are the specific approaches used in those stages.

(Refer Slide Time 28:35)



The material that I have covered in this lecture are taken from this book and you can refer to chapter 2, section 2.1 to 2.4.2 to get the material. So, that is all for this lecture.

Thank you and goodbye.