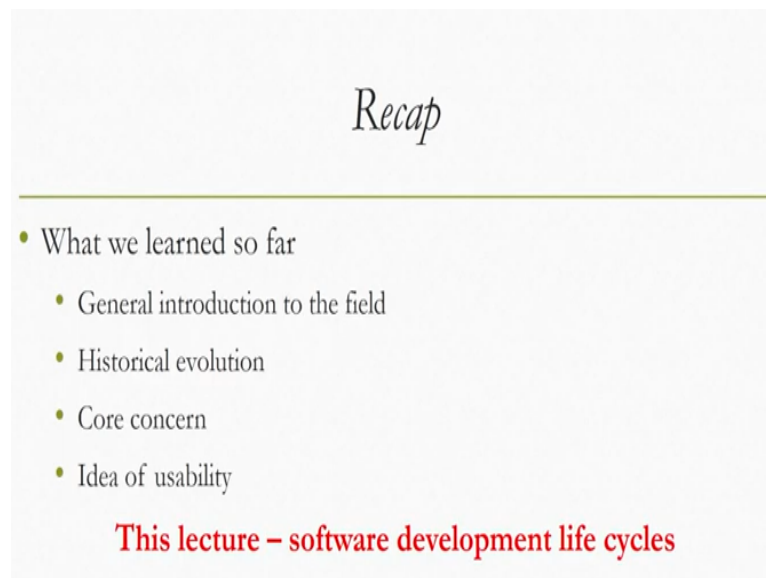


Design & Implementation of Human - Computer Interface
Dr. Samit Bhattacharya
Department of Humanities and Social Sciences
Indian Institute of Technology, Kharagpur

Module No # 01
Lecture No # 03
Engineering for Usability

Hello and welcome to the NPTEL MOOCS course on design and implementation of human computer interfaces. Today we are going to discuss our next topic that is engineering for usability this is lecture number 3 in this course. So before we proceed as usual let us first recollect what we have learned in the previous 2 lectures.

(Refer Slide Time: 01:12)



The slide features the word "Recap" in a cursive font at the top center. Below it is a horizontal green line. Underneath the line is a bulleted list with four items: "What we learned so far", "General introduction to the field", "Historical evolution", and "Idea of usability". At the bottom of the slide, the text "This lecture – software development life cycles" is written in red.

Earlier we have learned few things namely a general introduction to the field. So we have seen what interactive system is and what are the core challenges in designing interactive systems the issues and challenges? We have also briefly discussed the historical evolution of the field of user centric system design and development. And we have also looked at the core concern in the design of interactive systems namely the idea of usability.

So as I mentioned in my earlier lecture that our core concern here is to build interactive systems that are usable. So we have to take care of the usability of the system. Today in this lecture we are going to talk about how to do that? In fact, we will first learn the different concepts related to implement user centric design approach for usable system design.

(Refer Slide Time: 02:43)

Introduction

- User-centric software – core design concern
 - Design usable system (to cater to the needs and expectations of the “layman” users)

So our core design concern for user centric system design and implementation is design of usable system. Now why we need it? We need it to cater to the needs and expectations of the users remember. That here we are referring to layman users so our objective is to cater to the needs and expectations of the layman users. If we are able to do that then we can say with confidence that our system is usable.

(Refer Slide Time 03:19)

What We Need

- A “systematic” approach!

To achieve that what we need to do? We need to follow some sort of systematic approach. There should be some step by step process to build user centric software. So that at the end, whatever we get as output is a usable system.

(Refer Slide Time: 03:45)

Software Development Life Cycle

- To **comprehensively capture and represent** design and development activities

Now this brings us to the concept of software development life cycle so software development lifecycle or SDLC can be used to comprehensively capture and represent the entire set of design and development activities. So we have to perform some activities to design and develop a software now these activities together can be used to create a software development lifecycle. So that these activities can be performed in a systematic manner.

(Refer Slide Time: 04:33)

Engineering a Software

- Software development life cycles – **build software in stages**

So essential idea here is that we need to define stages of development for building the software. And software development life cycles are nothing but the stage wise development process followed to build a software.

(Refer Slide Time: 04:57)

Engineering a Software

- Example – let us try to create a calendar app
- **What should we do?**

Why this stage wise development is important? Because it gives us a way to systematically look at the problem of software development. Let us try to understand the benefit of a stage wise development process such as the one provided by software development life cycles with an example. Suppose we want to create a calendar app a simple calendar for say a mobile phone.

What we should do? If this problem is given to you, how you will start, where you will start, how you will proceed, these are some of the concerns that will immediately come to your mind.

(Refer Slide Time: 05:51)

Calendar App - A Possible Design

- 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)

First thing is of course we need to understand, what is required and then accordingly we should go for design of the app so what we need? A grid like structure that is typically found

on a physical calendar. So all of us are familiar with a calendar at many places probably we have seen it so there the dates are mentioned in the form of a grid similar thing we want to do in our calendar app.

Along with the dates so there are headings representing name of the month's. So that information also we need to put in the grid like structure.

(Refer Slide Time: 06:41)

Calendar App - A Possible Design

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

Now for each cell in the grid will represent a month. Now there are dates within a month so for representing the dates or days, we need to create sub grid for each cell in the higher level grid. So that is also required so create sub grids in each cell to hold the date or days. And finally we need to render the entire structure on the screen. These are our requirements which we have identified. Now to render it in the way we envisage the calendar there can be many possibilities .

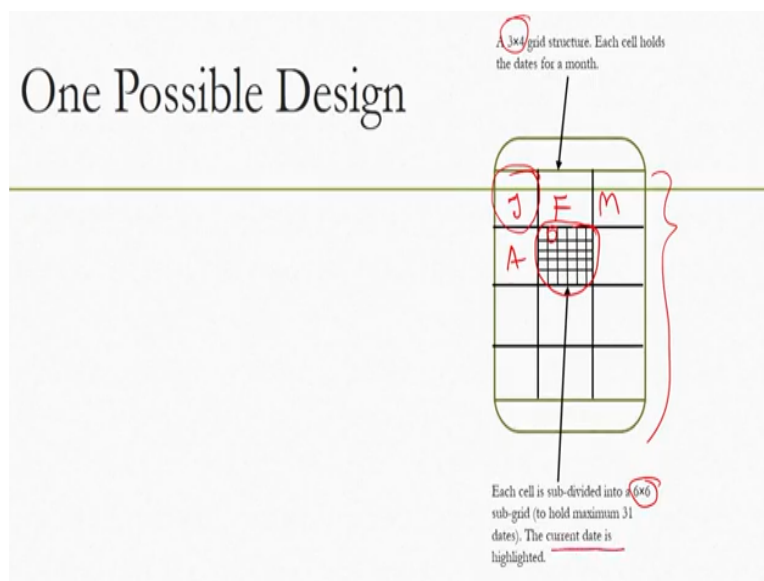
(Refer Slide Time: 07:30)

Calendar App - A Possible Design

- Highlight the current date and month

One more requirement can be there that is we may also highlight the current day and month or the current date. That may be an additional requirement we would like to have in our app.

(Refer Slide Time: 07:50)

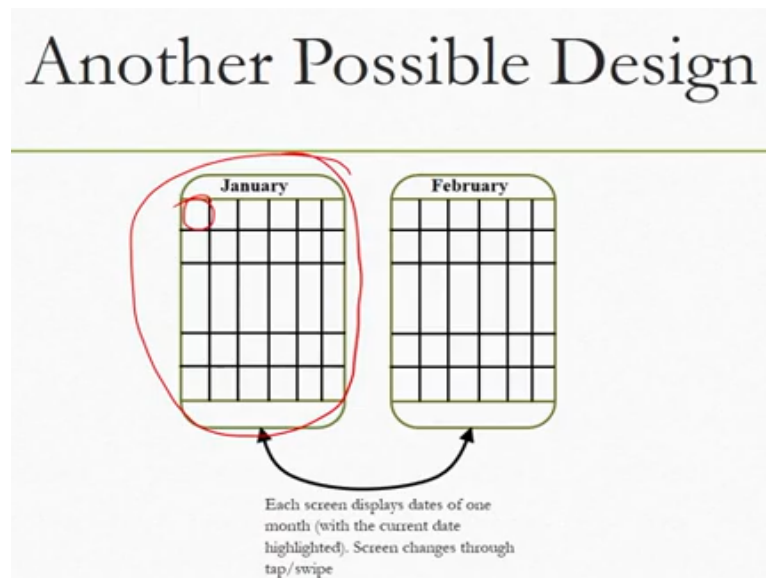


So there can be different ways to achieve this particular objective of having a calendar app. One possible design can be like this, so here we are mostly concerned about design of the interface how it looks like to the user? So we will mimic entirely the way physical calendars are there so there will be this grid like structure so maybe a 3 by 4 grid representing 12 months. So each grid element is corresponding to a month for example January, February, March, April and so on.

Within each grid element there can be sub grid elements representing the days of the month. Since in a month there can be at most 31 days so we can have a sub grid of 6 by 6 structure

now to be able to hold maximum 31 days. So each sub grid element then will represent 1 day of the month and in this scheme of things we can highlight the current date that is the day of the month. This is one way to do things there can be other possible, so there is no unique solution let us have a look at another way.

(Refer Slide Time: 09:43)



Now since we are building an app rather than a physical calendar we can utilize the flexibility provided by a computing system and, we can mimic another way of designing physical calendars, typically the tabletop calendars where we can change pages same idea we can implement here. So on one screen we will have days for a month with swipe or with tap I can go to the next or previous month like we change pages on a table top physical calendar.

So the entire screen will contain only the days for a particular month unlike in the previous case. And here we will have grid where each grid element will represent one day of the month. So with tap or swipe I can go to the next month or the previous month. And this scheme can of course be implemented by considering the fact that when I am going backward from January then I will end up in the last year or previous year. Whereas when I am going forward from December I end up in the next year.

So accordingly the days can be chosen or designed, so this is another way of designing things. In fact there can be a large number of alternative ways these 2 are not the only ways possible so the next question comes is then if there are so many ways then how we can actually decide which one to choose?

(Refer Slide Time: 11:50)

Design Alternatives

- There can be many possibilities
 - Challenge - how to choose the right one
- Require a systematic approach
 - SDLCs help

Our intuition may not be sufficient we probably have to take a systematic study of the pros and cons of different designs to find out the best one among the possible alternatives. And that systematic study is nothing but what we are referring to as software development life cycles. So given a set of alternatives, it is possible to follow SDLC's to determine the best alternative that is one advantage of having SDLCs.

Apart from that of course there is this added advantage, that we can look at the problem in a more systematic manner which will make things clearer and simpler to us to proceed. Otherwise if we are dealing with building large software then the things will quickly become messy if we do not look at it in a very systematic manner. And then it will be very difficult for us to come up with a usable product. Having identified the advantages of software development life cycles let us now learn a little bit on what are the different life cycles.

(Refer Slide Time : 13:10)

SDLC

- There are many
 - Waterfall model
 - Spiral model
 - Evolutionary model
 - ...

So there are actually many ways to look at the software development process accordingly several life cycle models have been proposed over the years such as the waterfall model, the spiral model, the evolutionary model and so on so we will have a look at some of these models to get a better understanding of what we are meaning by software development lifecycle models.

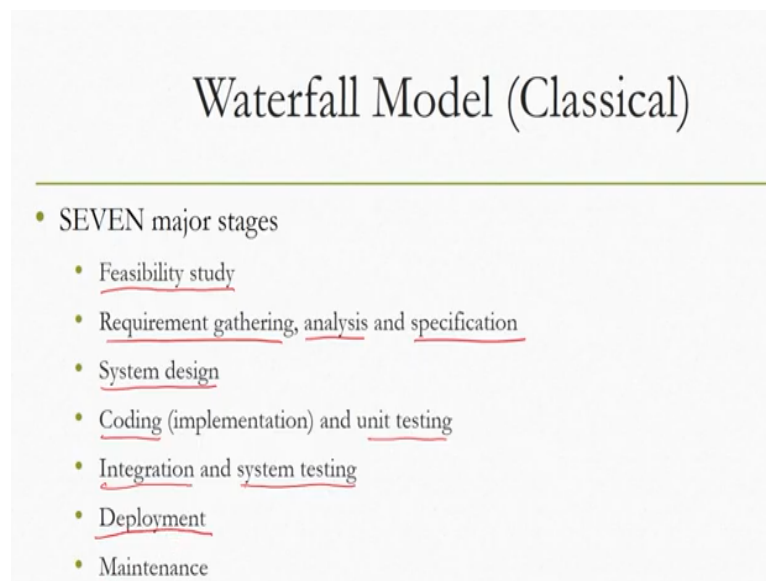
(Refer Slide Time : 13:46)

Waterfall Model

- Most well-known SDLC model (although rarely used in practice)

Let us start with the most fundamental of these models namely the waterfall model. This is also the most well-known SDLC model although this is primarily used for pedagogical purpose for teaching purpose rather than for actual software development. Now in this model remember that we talked about software development life cycle to be a stage wise process for building a software. Now in the waterfall model or rather the classical waterfall model there are 7 such stages which we should follow to build a software.

(Refer Slide Time : 14:21)



First is feasibility study before we start on the software development process, we first need to identify whether it is feasible whether we can proceed with the available resources, available budgetary support, available manpower and so on? So that is the first stage of the process. Where we first check for feasibility of the whole building process. If it turns out to be not feasible then it is preferable not to proceed further or we need to modify our goals accordingly based on availability of resources.

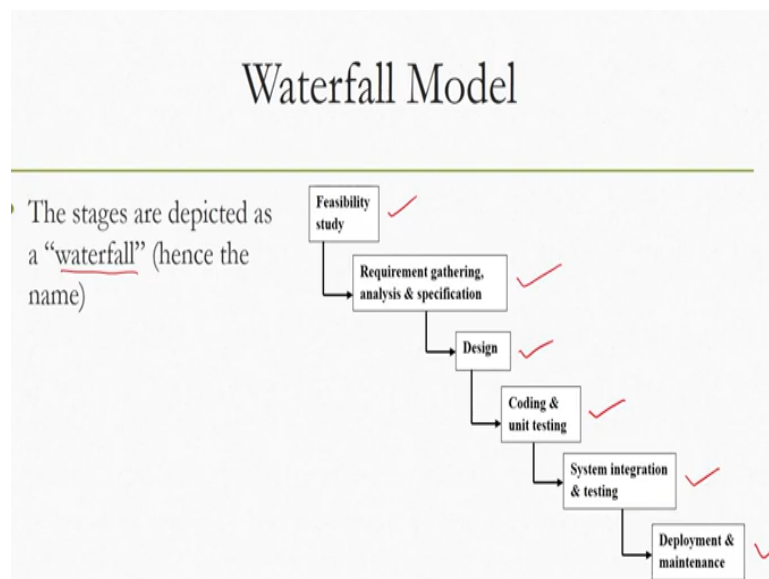
The second stage is requirement gathering analysis and specification. So we need to identify what is required to build the software. Here the requirement is not in terms of resources rather it is in terms of user requirements. So we need to identify and gather those requirements and analyze those to find out what is possible to be implemented? The third stage is the design stage here we go for designing the software.

Now later on we will see that the design involves 2 thing in the context of our focus. That is human computer interfaces here design refers to both interface design as well as the design of the system or code. Then comes coding or actual implementation of the system along with that we may need to test the code also for identifying bugs and taking corrective measures so coding and testing is the next stage.

Once unit level testings are done and codes are finalized. Then we go for integration of the unit levels and then overall system testing and the final stage is deployment of the product. It also includes maintenance as well which may be considered to be a part of this phase or can be a separate stage. Now these 7 stages in different places you may find that they are

mentioned in different ways so for the sake of discussion in this course we will assume that there are these 7 stages in the waterfall model.

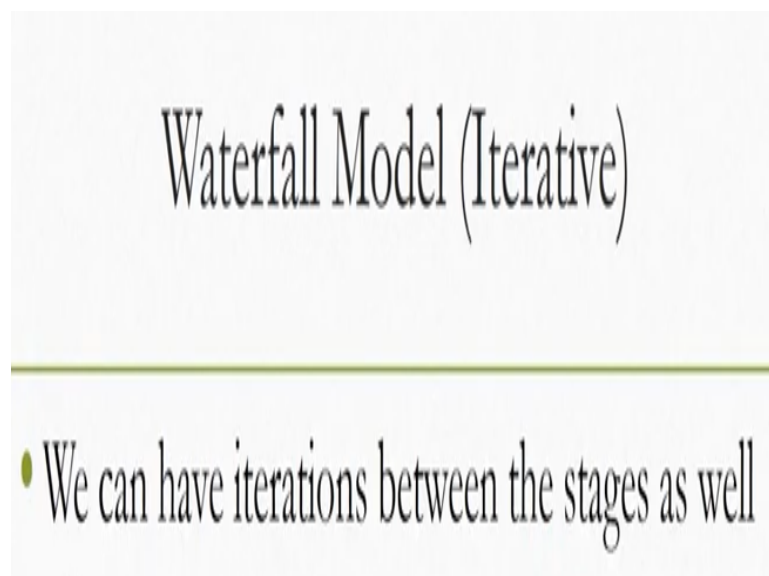
(Refer Slide Time: 16:48)



Now when these stages are depicted they are depicted as kind of a waterfall. As shown in this figure so first feasibility study next is requirement gathering analysis and specification. Third is design, then coding and unit testing system integration and testing deployment and maintenance. As you can see here either we can have 2 separate stages or single stage covering both.

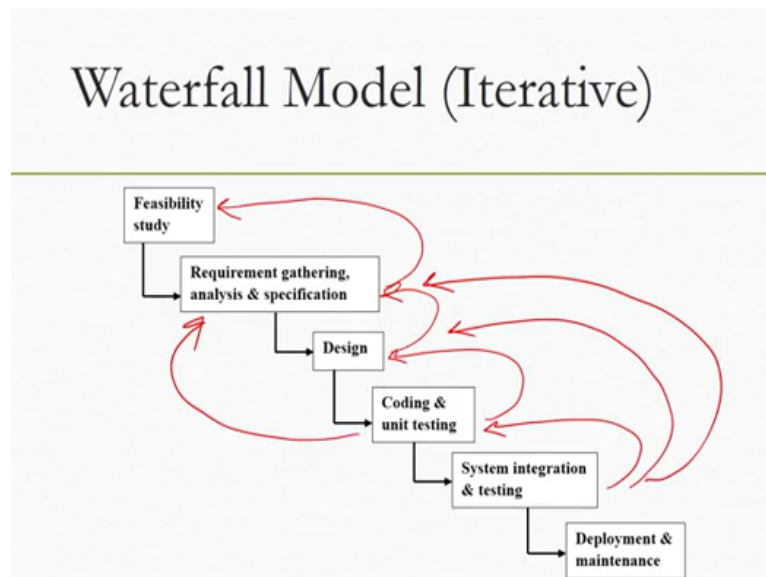
And the way it is depicted it appears to be like a waterfall so accordingly this name is given to this particular way of looking at the software development process.

(Refer Slide Time: 17:32)



Now that was the classical model. Now waterfall model can also have an iterative version where we can have iteration between the stages.

(Refer Slide Time: 17:45)



For example after feasibility study we go for requirement gathering analysis and specification. There we may find that few things are not available. Then we may like to go back to feasibility study. After design we may like to go back to requirement gathering stage. After coding we may like to go back to design stage or requirement gathering stage. Similarly after system integration and testing we may like to go back to coding stage or design stage or requirement stage anywhere.

So this type of iterations can also be accounted for in the traditional or classical waterfall model. If we are having iterations between the stages then it becomes known as iterative waterfall model. So that is the basic idea about waterfall model.

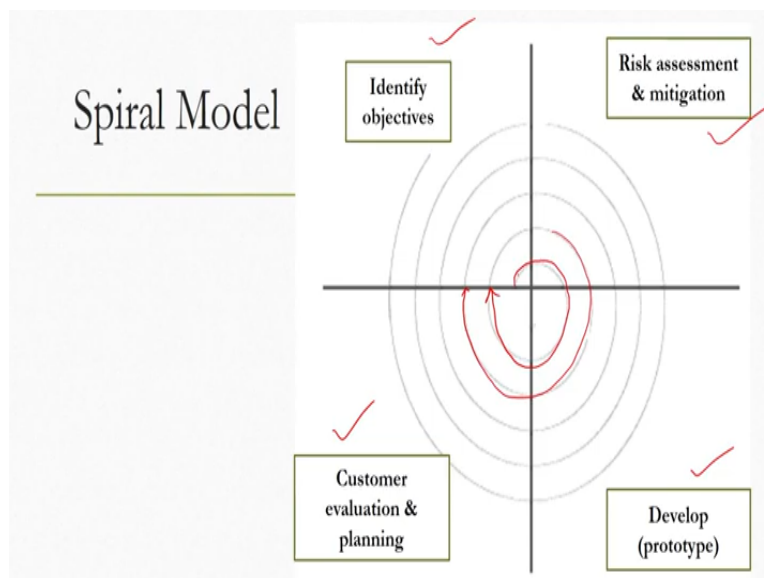
(Refer Slide Time: 18:59)

Spiral Model - Characteristics

- A meta model – encompasses other SDLCs

Another software development lifecycle model is the spiral model let us have a look at this model. Spiral model is actually a Meta model meaning it encompasses other SDLCs so it is kind of Meta level SDLC.

(Refer Slide Time: 19:10)



So when graphically illustrated the model looks something like this. So there are 4 quadrants identify objectives, risk assessment and mitigation develop prototype, customer evaluation and planning. Now as you can see here each spiral covers all the 4 quadrant. So starting from here it goes up to here that is one iteration then again it can start from here and till it covers the 4 quadrants that is another iteration so on. So there can be unlimited number of iterations as per the requirement of the development process.

(Refer Slide Time: 20:07)

Spiral Model - Characteristics

- Multiple cycles (iterations)
- Each spiral – one iteration
- Each iterations – divided into 4 phases (quadrants)

So there can be multiple cycles each cycle is an iteration, so each spiral is one iteration. Each iteration is divided into 4 phases or quadrants as depicted in the figure.

(Refer Slide Time: 20:28)

Spiral Model - Quadrants

- 1st quadrant – identify objectives & risks
 - During the first quadrant, objectives of the iterative phase are identified
 - Also the risks associated with these objectives

The first quadrant identifies objectives of the development of the product and the risks .So objectives of the iterative phase are identified in this first quadrant of the iteration and also risks associated with these objectives. These are also identified in this first quadrant.

(Refer Slide Time: 20:58)

Spiral Model - Quadrants

- 2nd quadrant - risk assessment and mitigation
 - Identified risks analyzed in details
 - Steps taken to reduce the risks
 - E.g, if there is a risk of inappropriate requirement specification, a prototype system may be developed

In the second quadrant that is risk assessment and mitigation so the identified risks are analysed in details. So in the first quadrant we have identified the risks those risks are analysed in details and then steps are taken to reduce the risks. For example if there is a risk of inappropriate requirement specification. Then a prototype system may be developed to gather requirements which will mitigate this risk.

(Refer Slide Time: 21:41)

Spiral Model - Quadrants

- 3rd quadrant - development
 - Develop product (prototype) after resolving identified risks and evaluate

Third quadrant deals with the developmental phase. So in this quadrant we can go for building or developing the product or the prototype. The earlier iterations and product was the latter iterations after resolving identified risks. Also evaluate the product or the prototype whichever is the case. So as I said there are multiple iterations so initially during the initial iterations typically prototypes will be developed and during the latter iterations, when we are almost coming to the end of the developmental process.

Focus will shift towards building of the product and during the iteration this product will also be evaluated. If it is product the product will be evaluated. If it is prototype then the prototype will be evaluated.

(Refer Slide Time: 22:48)

Spiral Model - Quadrants

- 4th quadrant – evaluation and planning
 - Review results achieved so far with customer and plan next iteration around the spiral

Then comes the fourth quadrant or the last quadrant. So here whatever results we have achieved so far after evaluation with customer in the third quadrant, those results are reviewed and we plan for the next iteration if required. So here we are primarily evaluating the results achieved so far and plan for the next iteration if required. So these are the four quadrants of a spiral model. As probably you have noticed by now spiral model is not a fundamental developmental model like the waterfall model.

Instead it captures higher level concerns and activities that are required to implement a lower level or fundamental level software development life cycle.

(Refer Slide Time: 23:54)

Spiral Model

- Through the iterations, progressively more complete version of the software gets built .

And through these iterations of the spiral progressively more complex versions or more complete version of the software gets built that is the overall objective. So with these we get some idea of what is the software development life cycle, so we discussed one fundamental SDLC model and 1 Meta model to get some idea of what we mean by stage wise or systematic development of a software.

(Refer Slide Time: 24:34)

Interactive System and SDLC

- Interactive systems should be “usable”
- Requires users to take into account in every stage of the design
- Brings in lots of iteration

Now our concern for building interactive software is that the software should be usable this requires the developer to take users into account in every stage of the design in one form or the other either in active form or in passive form. Now in order to do that if we go by straight forward approaches then that will require lots of iterations.

(Refer Slide Time: 25:15)

Interactive System and SDLC

- Difficult to express with traditional SDLCs (e.g., waterfall model)
- In the next lecture, we shall learn about an “iterative” SDLC for interactive systems

Now these iterations although not impossible but it creates difficulty if we want to use a traditional SDLC such as the waterfall model. That will create a problem as you probably have noted when I was talking about the iterative waterfall model. So there between every stages we probably need to iterate and it may be between 2 consecutive stages maybe far away stages to take into account user feedback properly and completely.

So the resultant model when visualized will look messy and it will be difficult to follow. If we follow the traditional models. In the next lecture we shall learn about an alternative model for building interactive software which is although takes care of the iteration. But it does not look messy as in the case of the waterfall model.

(Refer Slide Time: 26:45)

Reference

- Rajib Mall (2018). **Fundamentals of Software Engineering**, 5th ed, PHI Learning Pvt Ltd. **Chapter 2**
- Roger S Pressman (2015). **Software Engineering: A Practitioner's Approach**, 8th ed, McGraw-Hill Education, New York, **Chapters 2-4**

Whatever we have covered today you can refer to these two books to get more material on these topics Rajiv Mall Fundamentals of Software Engineering as and Roger Pressman Software Engineering of Practitioners approach. You may go through chapter 2 of the first book or chapters 2 to 4 of the second book to get a better understanding of the development process. And the process models that is all for this topic hope you have learned the basic idea will meet again in the next lecture thank you and goodbye.