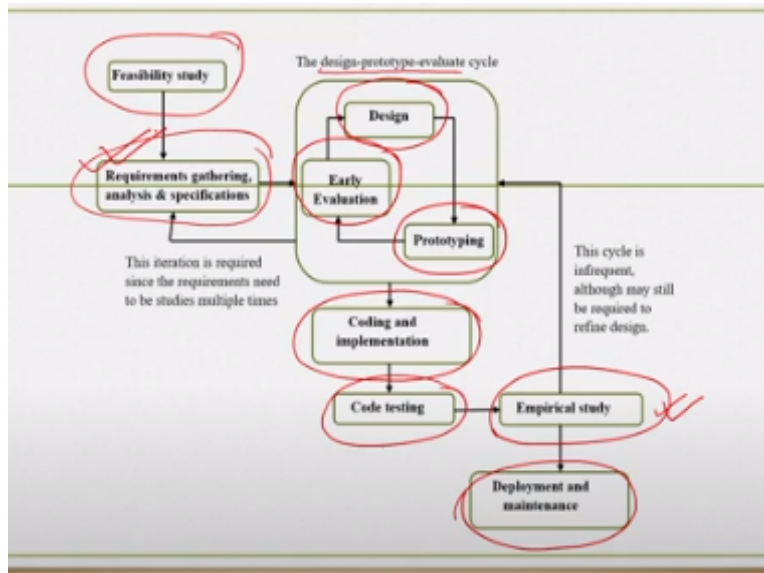


Design & Implementation of Human – Computer Interfaces
Prof. Dr. Samit Bhattacharya
Department of Computer Science & Engineering
Indian Institute of Technology, Guwahati

Module No # 03
Lecture No # 12
Introduction to Interface Design

Hello and welcome to NPTEL MOOCS course on design and implementation of human computer interfaces lecture number 11. In this lecture we are going to talk about the next stage in the interactive system development life cycle namely the design stage. Before we begin we will quickly recollect the life cycle stages that we are currently discussing.

(Refer Slide Time: 01:14)



So we have seen earlier one of the stages that is the requirement gathering analysis specification stage. Apart from this stage there are many more stages which together constitute the interactive system development life cycle. So we have feasibility study stage, then requirement gathering stage, design stage, prototyping stage early evaluation of the prototype that is another stage. Then coding and implementation stage, code testing stage, empirical study stage and finally deployment and maintenance.

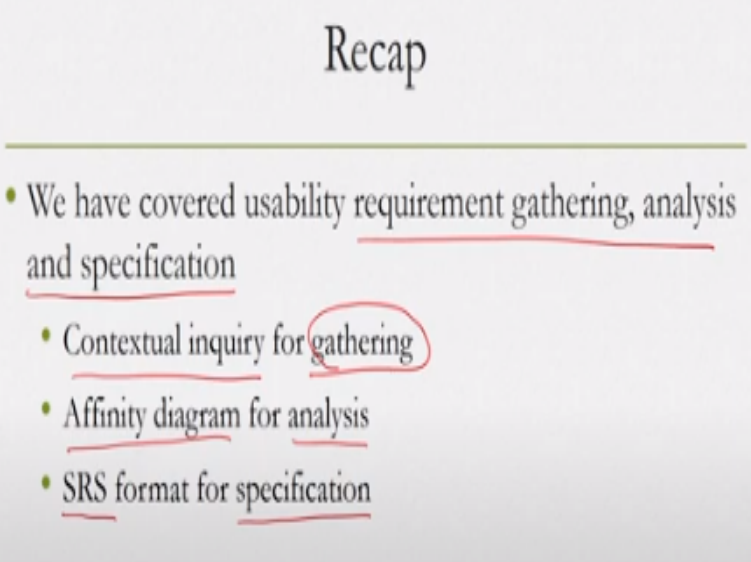
Now among these stages the relationships vary for example in the 3 stages design prototype and early evaluation they are connected in the form of a cycle. So we first design then create a

prototype then evaluate the prototype based on the evaluation results we refine the design create again another prototype based on the refined design again evaluate and this goes on in a cycle. So this cycle we are calling as design prototype evaluate cycle.

The cycle stops when the design stabilizes that means no further problems are found in the design after the evaluation of the prototypes. Now this design prototype evaluate cycle primarily refers to the design of user interfaces apart from that the design stage also refers to the design of the code or the program which implements the design. So both designs are referred to by this design stage and in code testing we test the code for its ability to execute properly along expected lines.

Whereas in empirical study we test usability of the end product by involving end users in the evaluation of the whole product or software. So these are the stages and through these stages we try to build one interactive system that is executable as well as usable. Among them we have already discussed one stage that is the requirement gathering analysis and specification stage.

(Refer Slide Time: 04:04)



Recap

- We have covered usability requirement gathering, analysis and specification
 - Contextual inquiry for gathering
 - Affinity diagram for analysis
 - SRS format for specification

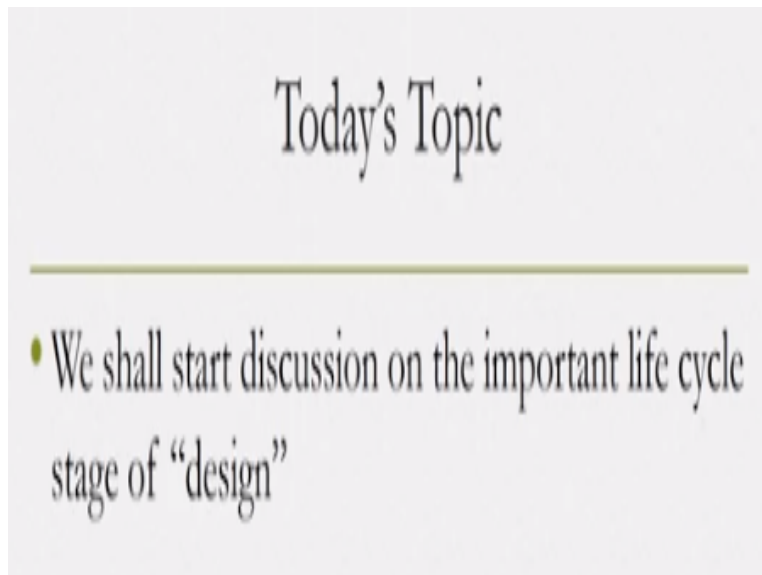
So in this stage what we have learned? We have about the idea of usability what is meant by the term usability requirements? Also what are the other requirements from the point of view of the system development? So the usability requirements are non-functional requirements and we have also learnt functional requirements and learned how to create functional requirements from usability requirements through one case study?

More specifically what we have learned are usability requirement gathering and one method. So there are several methods for gathering usability requirements we have learned one particular method that is contextual inquiry. Then we have learned the affinity diagram method which works on the data collected through contextual inquiry to analyze the results of the gathering exercise. So we learnt affinity diagram method for analysis of the requirements that we have gathered during contextual inquiry.

Thirdly we have learned how to specify the requirements so we have learned about SRS or software requirement specification document how to create it what are the conventions to be used and so on which is required for specification of the requirements. So we covered the requirement gathering analysis and specification stage by separately learning what you mean by gathering what you mean by analysis and what we mean by specification?

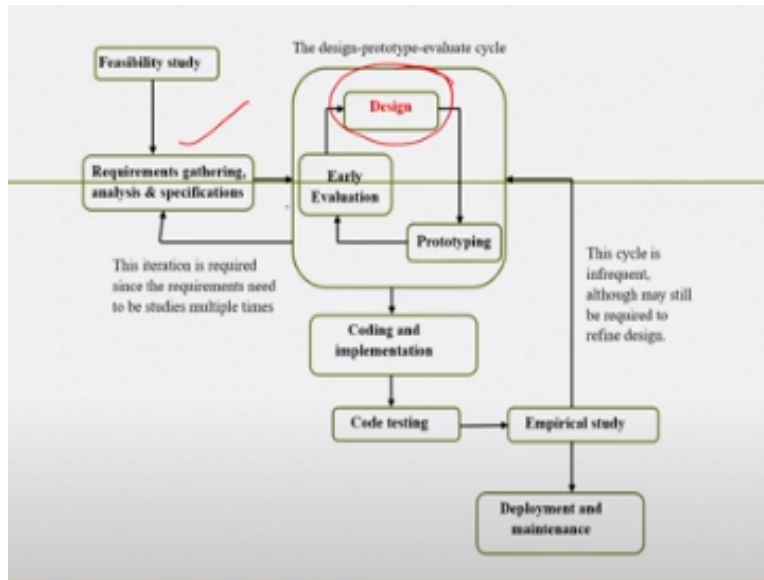
Whatever we have learned in this stage we try to explain that in terms of one case study of an application namely a calendar application. Primarily to be used by the student community who are enrolled for higher studies in higher educational institutes including colleges and universities and who are supposed to use it for academic purposes.

(Refer Slide Time: 06:30)



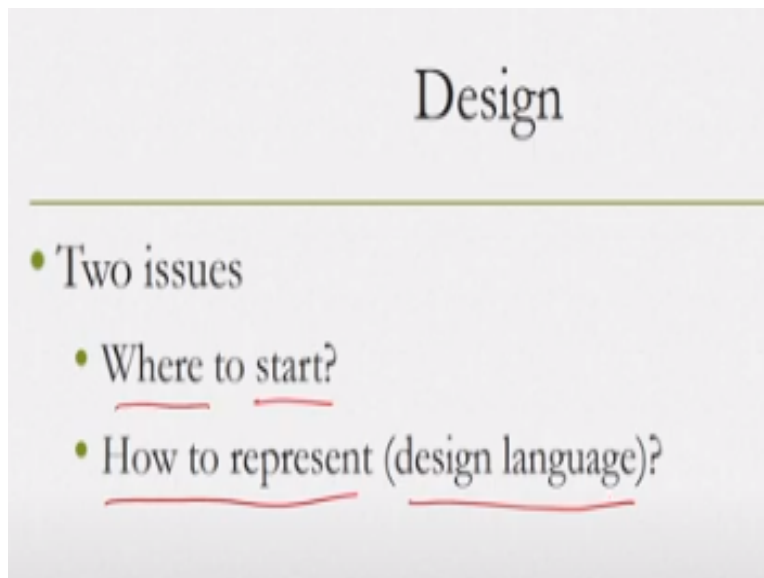
In this lecture we are going to talk about the next stage that is the design stage which is one of the components in the design prototype evaluates cycle.

(Refer Slide Time: 06:42)



To recollect so we have covered this stage and now we are going to talk about the design stage subsequently we will talk about the other stages namely prototype and early evaluation and how the cycle is formed. So in this lecture we will get a general introduction to the idea of design in a software development life cycle. So what we mean by design and what we should know and take care of in the design process? So when we talk of design what are the primary concerns?

(Refer Slide Time: 07:21)



There are 2 major issues or broadly 2 issues that we should be aware of and we should try to address in our design activities. What are those issues? The first most obvious issue is where to

start the process? Where to start so; essentially when we are asked to come up with a design we should be very sure or we should be knowledgeable about the starting point.

Where to start the process otherwise it may lead us to nowhere and we may end up with a faulty or incomplete design. The second major issue is once we have designed something the concern is how to specify it how to represent it? So that later on in later stages when we go for implementing the design or go for doing something else on the design we can communicate the idea of design clearly to the other members of a development team.

So the second major issue is how to represent our design in other words which language to use to represent the design. So there are these 2 major issues that we should be concerned about when we are going to work on design. First one is where to start and second is how to represent both are very important considering that the entire life cycle is a team effort. So; one output of one stage should be passed on to the input of another stage which may be executed by a separate group, separate person or separate members of the team.

So this input that the team gets which is the output of some previous stage should be clearly understandable. So that the new member can start working on the idea without any problem without requiring to, consult the member of the team which executed the previous stage that is very important. And from that point of view we should keep in mind that communicating ideas from one stage to another in a very clear and precise manner is very much important and part of this overall exercise. So we should always be concerned about that issue how to represent and where to start.

(Refer Slide Time: 10:07)

Where to Start?

- Creative thinking!
 - May be aided by intuition
 - May be aided by experience/domain knowledge

Let us now start with the first issue that is where to start where we should start the design activity? Obviously when the term design is mentioned it implies that we have the freedom to think in our own way. In other words we should have the freedom for creative thinking. So as a designer we should be able to creatively think and come up with a design now that is one way of looking at the issue.

Now creative thinking need not work in a vacuum so when we say we are trying to come up with a design through creative thinking. Implicitly we may be relying on our intuition or our past experience both may be aiding in our thinking process. So effectively when we are trying to think in a creative manner we may be aided by the intuition which we may have and our past experience or knowledge of the domain of the problem that actually influences our creative thinking.

(Refer Slide Time: 11:26)

Where to Start?

- In UCD (user-centered design), we have TWO concerns
 - Interface design ✓ user
 - Code design ✓ system

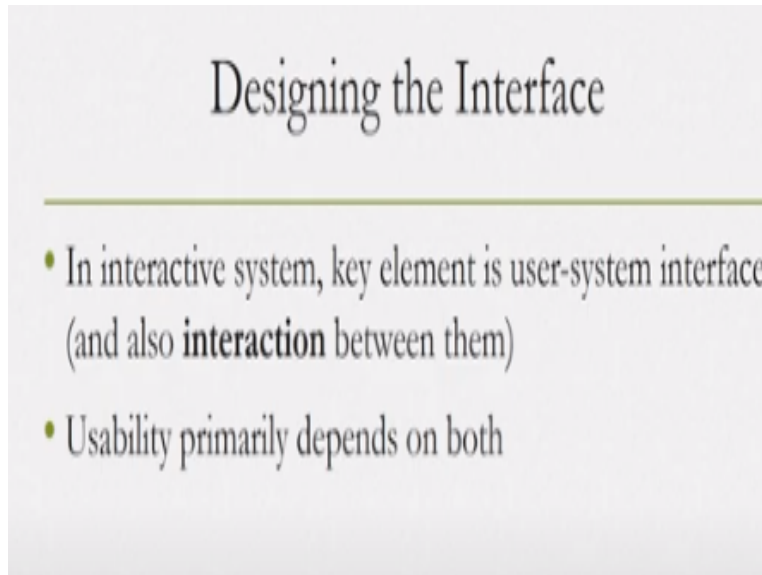
Now; that is a broad way of looking at things now let us come to a bit more specific issues. So when we talk of UCD or user center design now here the design is not a single idea here design actually encompasses 2 distinct concepts. One is interface design that means the user interface that the user gets to see and interact with. Now interface design also includes design of the interaction through with this interface.

That means essentially interface plus interaction design that is one aspect of the design and second is the design of the system or code design. So we have these 2 concerns interface design which includes interaction with the interface as well now this is primarily from the point of view of the user. And code design which is primarily point of view of the system so both are important for getting or for developing an executable and usable product so how to take care of these 2 concerns?

Let us first start with the idea of interface design to recollect when we talk of interface design essentially what we are referring to? We are referring to the user interface. That means the system interface through which the user gets to interact with the system. Now the term interface implicitly contains the idea of interaction as well. So without interaction interface is not of much use so when we say interface design we are actually referring to both designs of the interface as well as the corresponding interaction.

So when we are concerned about design of the interface we should be very careful because our primary concern here is usability and usability primarily depends on the design of the interface and interaction.

(Refer Slide Time: 13:54)

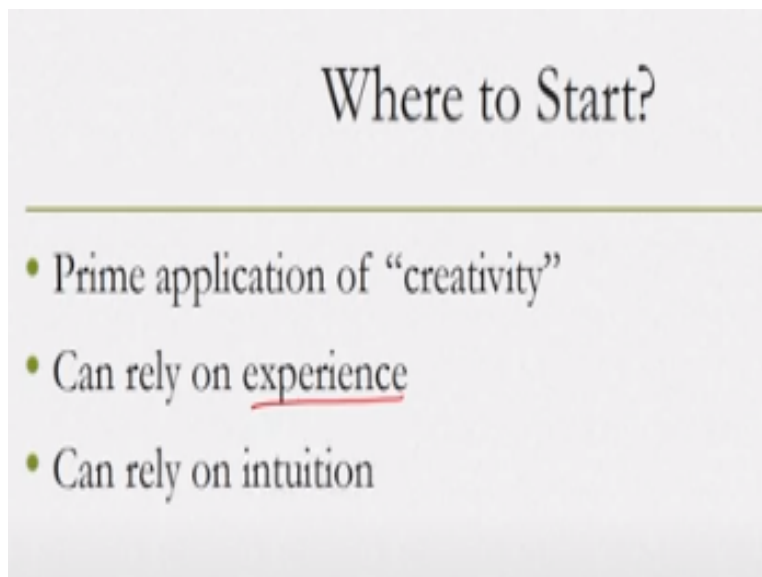


Designing the Interface

- In interactive system, key element is user-system interface (and also **interaction** between them)
- Usability primarily depends on both

So these 2 affects the usability of the product because these 2 are the portal through which user gets to view the system perceive the system and achieve goals with the system. So here the main concern is to come up with a design that increases usability of the product. Now let us come back to the core issues of design where to start and how to specify?

(Refer Slide Time: 14:25)



Where to Start?

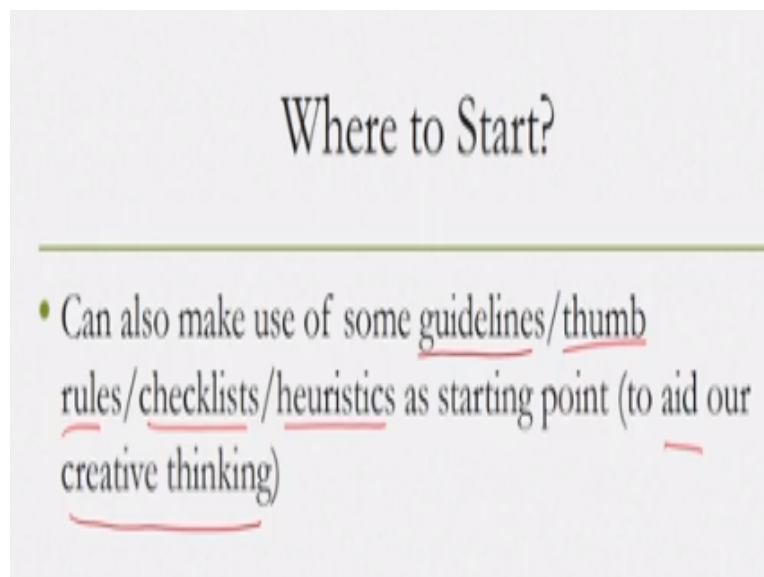
- Prime application of “creativity”
- Can rely on experience
- Can rely on intuition

So when we are talking of interface design where to start so we can start with creative thinking as we have seen before. So this is actually a prime application of creativity. How the interface should look like? How the interaction can be designed? How the interface elements should be designed? How they should be placed? Creative thinking can aid us in making all these design decisions.

And as we have noted earlier so this creativity can be added by experience so the designer may have experience of designing similar products before. Now that knowledge that experience can be used can be an influencing factor for coming up with a creative design idea. The creative thinking can also be aided by our intuition but of course intuition again is a very complex concept and intuition comes with knowledge with experience with wisdom.

So everything actually influences the way we tend to think we tend to create so the more experienced we are it will be easier for us to come up with a creative design solution the less experienced it will be difficult to come up with a creative solution. So there is no ready-made rule to tell you how to come up with a creative solution it entirely depends on your experience intuition and creativity.

(Refer Slide Time: 16:21)



However creative thinking is a specialized way of looking at things but there are methods that are easier to understand and access rather than having exotic skills like creative thinking. What we can do instead is? We can make use of some guidelines or thumb rules or checklists or

heuristics as starting point of design these can also aid in our creative thinking. So that is something very important to note that there are guidelines framed to aid designers in better design better interface design.

Similarly there are checklists created to aid designers ensure improved interface design there are heuristics which have been developed to help a designer come up with a good design and there are thumb rules. So; all of these terms actually refer to a set of guidelines or principles using which we can create quote, unquote good design which is likely to be usable. Now why we say that when we follow a guideline we can come up with a usable design because of the history of these guidelines or checklists or heuristics or thumb rules whatever you call it.

These are developed based on a long history of application of such guidelines rules or checklists on building usable products. So based on the knowledge gain these sets were continuously revised updated so that at the end whatever they contain is proven to be useful for building usable systems. Accordingly we can claim that whatever design we come up with based on these guidelines or heuristics can lead to improved usability enhanced usability.

That is one way of looking at the issue of where to start so we can start either as a creative thinker, creative designer or we can start with some set of guidelines which will aid us further in our thinking process.

(Refer Slide Time: 19:21)

Design Guidelines

- Many guidelines are there

Now there are many guidelines which have been developed over time for various purposes.

(Refer Slide Time: 19:29)

Design Guidelines

- Some very generic and consequently small in size - covers broad aspects of interactive systems, at a rather high level
 - ✓ “eight golden rules” (Shneiderman, 1986)
 - ✓ “seven principles” (Norman, 1988) WIMP

Some guidelines are very generic in nature that means they pertain to very broad aspects of a design of a usable design and because they are very generic. So they are usually small in size that means the number of guidelines is less in value maybe five to ten guidelines that constitute a set of guidelines for design of a particular system. So these guidelines because they are generic usually cover broad aspects of interactive systems at a rather high level again there are many such guidelines.

For example the 8 golden rule are the 7 principles the first one 8 golden rules was proposed by Ben Shneiderman, way back in 1986 the second one 7 principles was proposed by Donald Norman in 1988. Now both these sets actually originated during the proliferation of graphical user interfaces. You may be familiar with this term graphical user interfaces these are the interfaces that contains windows, icons, menus and pointers or what we generally call wimp interfaces.

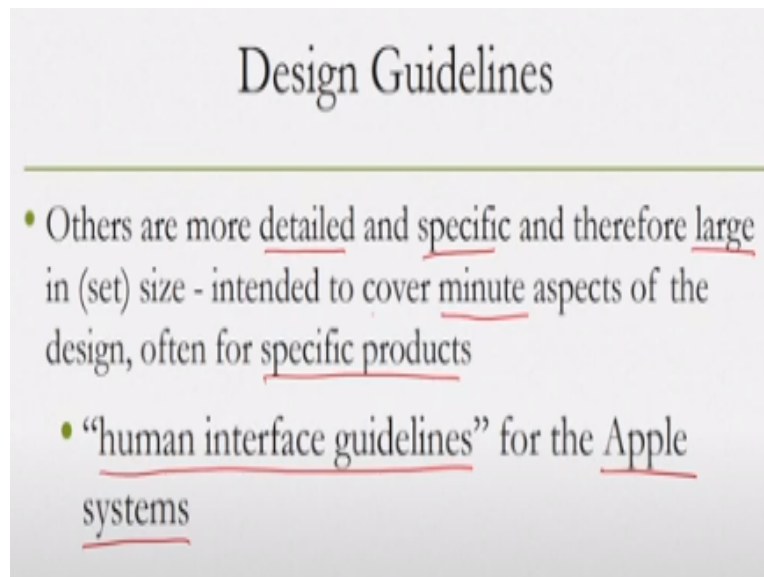
Windows, icon, menus and pointers or wimp interfaces these were ubiquitous interfaces at one time still they are very much around us. Whenever we interact with a computer desktop computer in particular or laptop whatever interfaces we get to see are mostly wimp interfaces. So these interfaces were developed way back in nineteen eighties at that time what we take for

granted nowadays was not so in at that time and the idea of usability was not that much developed.

At that time these golden rules or principles were developed by the corresponding persons in the case of golden rules it was Ben Shneiderman and in case of 7 principles it was Donald Norman. So these were developed to help in building (())(22:21) that are going to be usable. So that was the background of these guidelines and another thing that we should keep in mind is that these guidelines did not appear suddenly out of nowhere.

So they were framed based on empirical data collected over a long period of time and after analysis of that data some behavioral models were formed and out of those models these guidelines emerged. So they were rooted in behavioral analysis of human users so whatever guidelines are proposed if we follow them then we are likely to get systems that will help the users use those systems now these are some generic examples of guidelines.

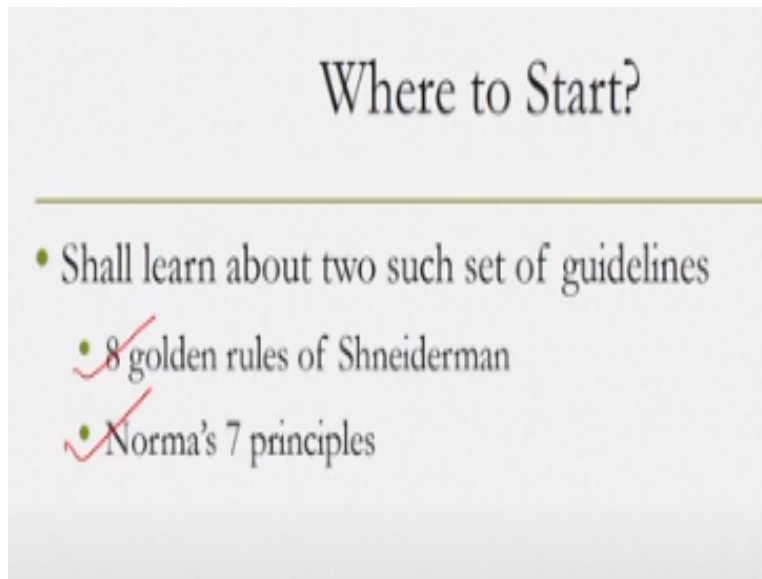
(Refer Slide Time: 23:30)



There are more detailed guidelines as well which are more specific and accordingly these are large sets that mean each set of guidelines contains hundreds or maybe even thousands of individual guidelines. Now these guidelines the detailed ones are intended to cover very minor aspects of the design and most often they are specifically meant for specific products.

One example can be the human interface guidelines that was developed for the apple systems this set of guidelines contained hundreds of individual guidelines which are primarily meant to build products on apple platforms on apple OS. And primarily products that deal with GY'S on apple platforms.

(Refer Slide Time: 24:36)

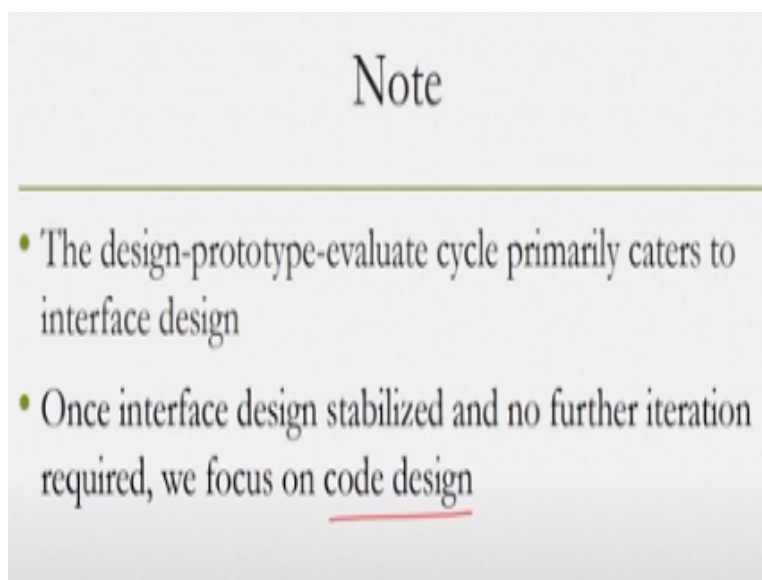


Where to Start?

- Shall learn about two such set of guidelines
 - ~~8~~ golden rules of Shneiderman
 - ~~7~~ Norma's 7 principles

In subsequent lectures we are going to learn about the 2 guidelines that I mentioned earlier eight golden rules of Shneiderman and Norma's 7 principles to get some idea on what we mean by the guidelines in this case the generic guidelines.

(Refer Slide Time: 24:57)



Note

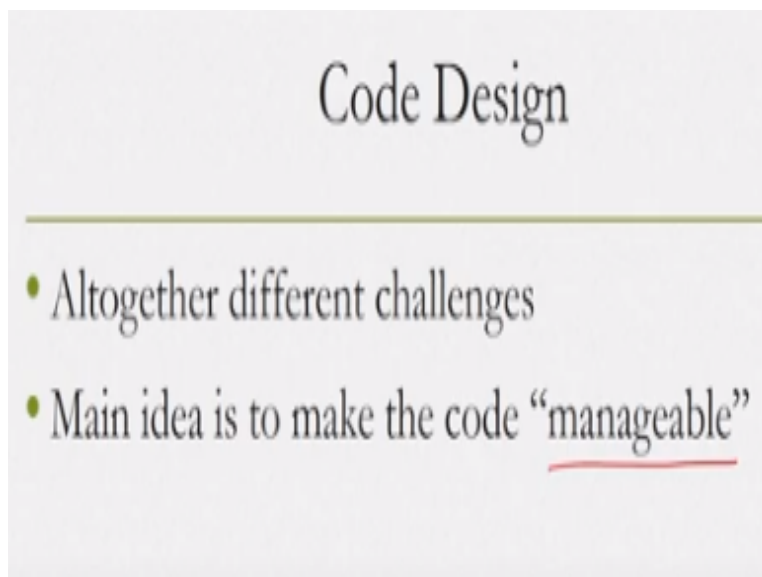
- The design-prototype-evaluate cycle primarily caters to interface design
- Once interface design stabilized and no further iteration required, we focus on code design

So that is about where to start now the guidelines help us to start design but design of what. So here we talked of 2 types of design one is interface design other one is code design. Now the guidelines are helpful to get us a starting point for interface design the other aspect of the design stage is the code design for which we need to take different approach so for code design let us see what we can do?

To repeat when we reach the code design stage once, we complete the design prototype, evaluate cycle and stabilize the design of the interface. So first we start with interface design we design something prototype it, test it if some issues are found out then we refine the design again prototype is test it. So this cycle goes on till we reach a design which is stabilized that means no further issues are found in the design.

In that case we move to the next aspect of the design stage that is code design so now we move to design of the system or code. Now code design is totally different than interface design.

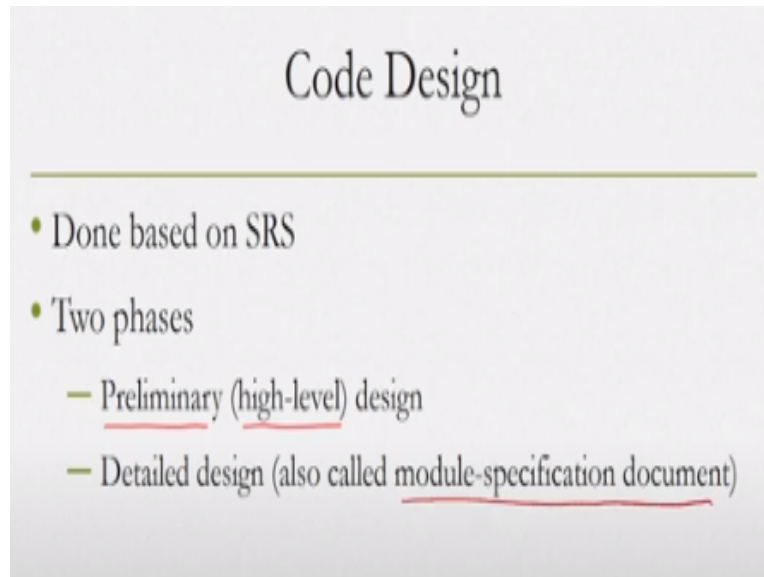
(Refer Slide Time: 26:20)



In interface design primary concern was usability and the design guidelines were again based on users behavioral characteristics and they were primarily meant to help us design usable products. But those are not useful in code design where our primary focus is efficiency of the implementation. Along with that our primary focus is to make the code manageable. That means we come up with a design which other team members can understand and manage if required modify also so that it can be implemented.

So the objectives changes and accordingly we should change our approach to address the main concerns.

(Refer Slide Time: 27:11)



Now earlier in the requirement stage we saw that we need to create an SRS or software requirement specification document as the end product of the stage. Code design actually is based on the SRS that we create so we need to translate the functional requirement hierarchy to the design of a system that can be implemented. So when we talk of code design there are 2 distinct phases are involved the first one is the preliminary phase or high level design phase.

And there is another one that is the detailed design phase also called module specification document. So; any code design activity involves these 2 stages high level design and or preliminary design and detailed design or module level specification document.

(Refer Slide Time: 28:24)

High-Level Design

- Identification of modules
- Control relationships between modules
- Definition of interfaces between modules

So in the high level design what we need to do we need to identify the modules? That means the units which together constitute the overall system. Then define relationships between the modules control relationships between the modules how the modules are linked to each other. And also we need to define interfaces between modules how the modules talk to each other so how they are connected and how they are able to talk to each other?

(Refer Slide Time: 29:12)

Detailed Design

- Identification of data structures and algorithms for different modules

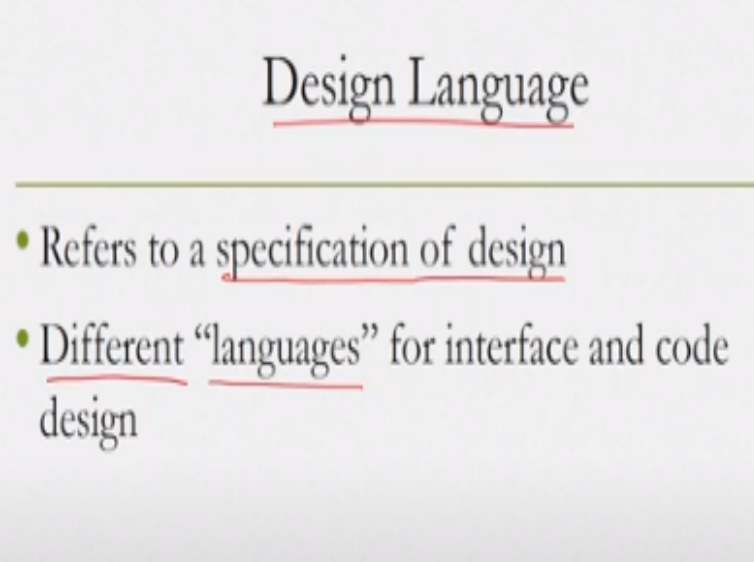
So all these things we need to consider in the preliminary design phase or high level design phase in the detailed design phase. We turn our attention to the design of individual modules

particularly the data structures and algorithms to be used to implement individual modules. So that is the second phase of the design activity code design activity that is detailed design phase.

So we started with 2 issues broader issues for any design activity one is where to start and how to represent? Now earlier we saw that in interactive system design there are 2 types of design one is interface design other one is code design. For interface design starting point can be creative thinking or it can be use of standard guidelines. Another thing we should note here is that the second issue that is how to represent the design? In case of interface design the representation is done with prototypes.

So prototypes effectively give us a language to express the design we will learn more about prototypes later. In case of code design our starting point is this phases, preliminary design phase and detailed design phase in preliminary design phase we work with high level concepts namely what are the modules broader modules? How they are connected? How they talk to each other? And in detailed design phase our primary concerns are the data structures and algorithms to be used to implement each module. Now let us shift our attention to what are the languages using which we can specify our designs.

(Refer Slide Time: 31:12)



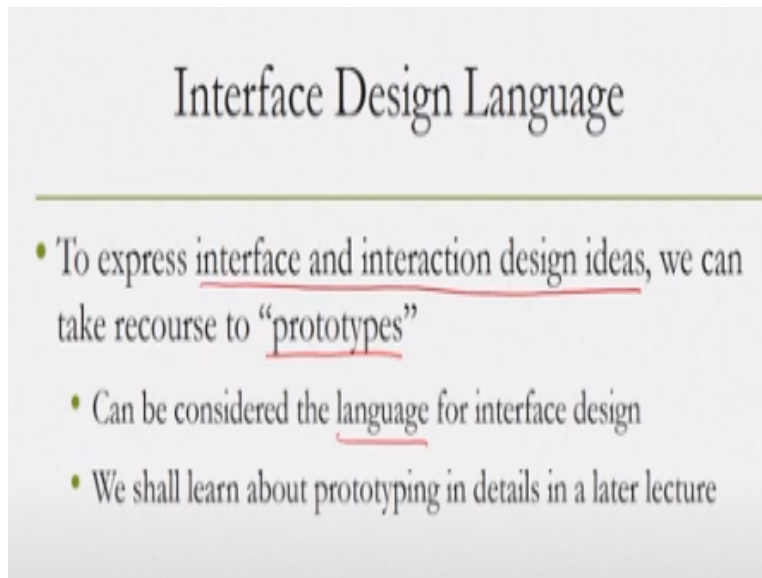
Design Language

- Refers to a specification of design
- Different “languages” for interface and code design

Now as briefly mentioned so when you talk of design language; we refer to specification of the design. Like we have seen specification of requirements using SRS which is a convention in

other words we can call it a language. Similarly we need a language to specify our design and we require different languages for interface and code design specifications.

(Refer Slide Time: 31:44)



The slide is titled "Interface Design Language" in a serif font. Below the title is a horizontal line. The main content consists of three bullet points:

- To express interface and interaction design ideas, we can take recourse to "prototypes"
- Can be considered the language for interface design
- We shall learn about prototyping in details in a later lecture

Again as I just mentioned earlier to express interface and interaction design we do not do it using natural language like English or Hindi or similar languages. Instead what we do is we make use of prototypes we take records to prototypes. Prototypes are scaled models of the designs when we create prototype we essentially express the design. So prototype gives us the language to express our design so we can consider a prototype to be a language for expressing interface design.

Now in a later lecture we will learn more about what are the different types of prototypes and how to create those prototypes? Clearly we can see here that prototypes although we are calling it a language but they are not a very formal language as such it does not have alphabet or anything. Rather we are using the term language here in a very loose manner whichever allows us to express something we are calling it a language. And in that sense prototype can be considered to be a language to express interface design.

(Refer Slide Time: 33:11)

Code Design Language

- Code design can be expressed in various ways
 - Natural languages
 - Semi-formal languages
 - Formal languages

How we can express the code design whatever we have designed through preliminary and detailed design phases how we can express that. So what is the language we can use for expressing system design or code design? We can actually take records to various ways so there are different ways to specify a system design. Definitely we can make use of natural languages although it will lead to ambiguity.

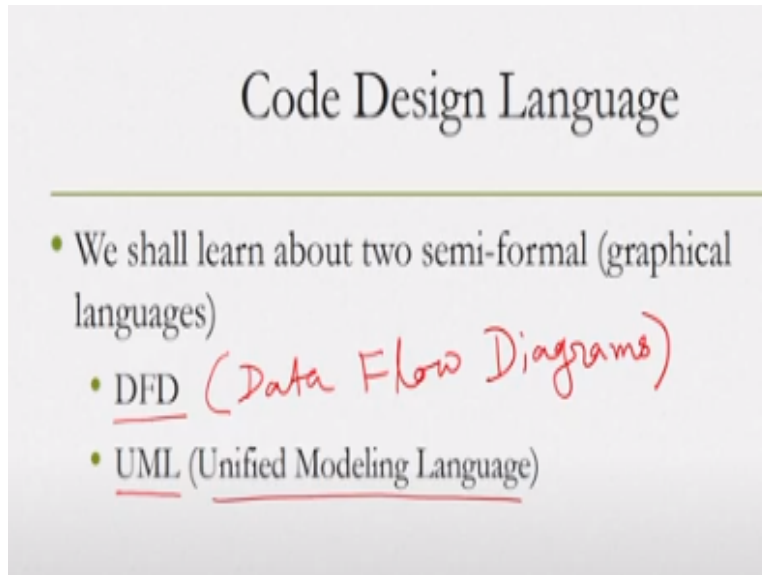
We can make use of semi-formal languages and finally we can go for formal languages with clearly defined syntax and semantics which are mathematically defined. So each has its own pros and cons if we are making use of natural language to express a system. Now that is very much subjective. So when I create a description of the system in terms of a natural language I may be using my literary skill to express it.

But later on when I pass on the message to somebody else when I pass on the design to somebody else in my team the particular member may not be able to comprehend it or maybe having lots of confusions because natural languages are inherently ambiguous. So natural languages although can be technically used but are not suitable or preferable to express design languages. This leaves us with 2 choices semi-formal languages and formal languages.

Formal languages are the best way to express any design however because it is a very rigorous expression it requires mathematical background. And good skills in understanding the particular notations and conventions and also understanding the logical deductions that follow from the

specifications. A middle ground can be found with a semi formal languages based specifications which are natural language. Partly graphical language and rather informal way of expressing things and easier to understand but at the same time it reduces ambiguity.

(Refer Slide Time: 35:53)



Code Design Language

- We shall learn about two semi-formal (graphical languages)
 - DFD (Data Flow Diagrams)
 - UML (Unified Modeling Language)

In this course we will focus more on this semi-formal language based specification of design particularly system design. So we learn 2 such languages one is DFD or data flow diagram which is a graphical language to express design in the form of drawings images. Second one is also a partly graphical language where we can express design in the form of some schematic diagrams this is called UML or unified modeling language.

First one is DFD or data flow diagrams as the name suggests it contains diagrams to express design which makes it easier to understand and makes it easier to maintain as well. Second one is UML or unified modeling language this is again partly graphical language of course in both the languages natural language text is also used along with the graphical notations. So these 2 languages we will learn in subsequent part of this course.

(Refer Slide Time: 37:29)

Note

- In the next lecture, we shall learn about design guidelines

So with that we come to the end of this particular lecture so here we got an overview of what we mean by design? To recap we are now going to start our discussion on the design phase of the life cycle now there are 2 types of designs are important one is interface designs other one is code design. Code design is also referred to as system design so we will use the terms interchangeably in subsequent lectures.

Now in both the cases what we should be aware of is what we should be concerned about are 2 issues one is where to start the design process and second one is how to specify the design? In case of interface design starting point typically considered to be creative thinking aspects but we can also start with the help of some standard guidelines. In case of code design we start at the preliminary design phase and also complete the design in the detail design phase.

Now the specification languages differ for each type of design in case of interface design we can consider prototypes to be a language quote, unquote language to specify the design. And in case of system design we can consider different languages however we will focus on semi-formal languages to specify the designs. 2 semi-formal languages will learn one is DFD or data flow diagram and other one is UML or unified modeling languages.

In the next lecture we will start our discussion on the design guidelines these are the guidelines that are useful to start design of interfaces. That is all for this lecture I hope you managed to learn

the concepts and enjoyed the content looking forward to see you all in the next lecture thank you and goodbye.