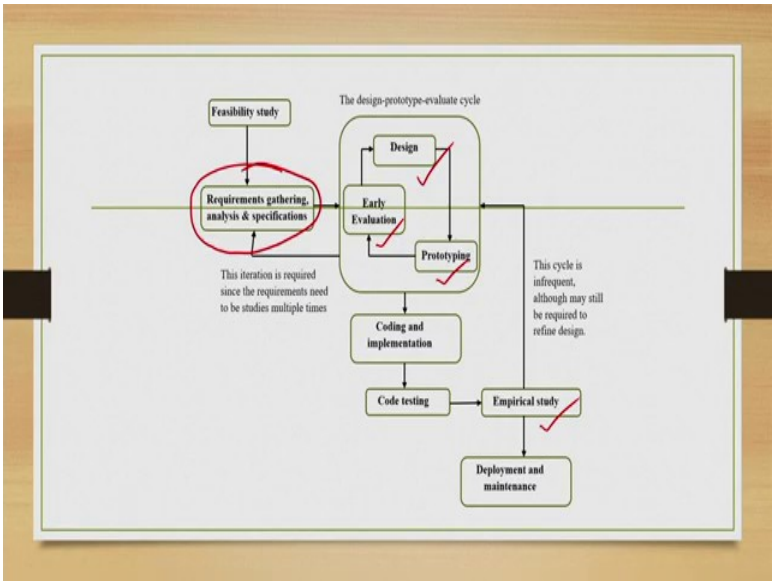
User-Centric Computing for Human-Computer Interaction
Prof. Samit Bhattacharya
Department of Computer Science & Engineering
Indian Institute of Technology, Guwahati

Lecture - 06
Components of SDLC - Design Guidelines

Hello and welcome to the 6th lecture in the course User-Centric Computing for Human-Computer Interaction. So, before I go into the details we may briefly recollect that we are now discussing different stages of the interactive software development lifecycle. In the last lecture, we have discussed the requirement gathering stage and one more stage we will discuss in this lecture.

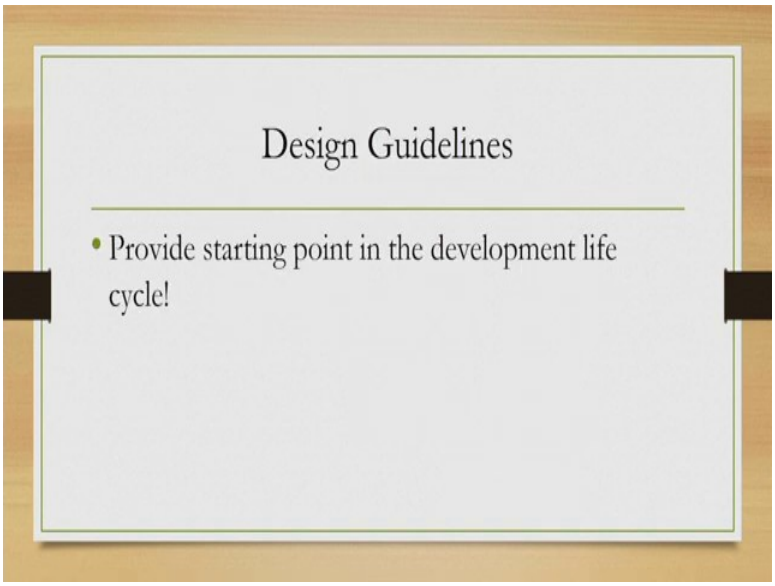
(Refer Slide Time: 01:05)



Now, just to help you recollect the interactive development lifecycle, this is the life cycle. As you can see in the figure there are several stages; we have already discussed one method to perform the stage namely the contextual inquiry method, other important stages are design, prototyping, early evolution, empirical study.

So, these are some stages that are unique in the context of or that are done in a different way in the context of interactive systems and in this course, we learnt different ways to do this. So, right now we are going to discuss the design stage, which is a stage that is done somewhat differently than a typical design stage in non-interactive software.

(Refer Slide Time: 01:51)



So, when we say we want to design something, remember that our objective in user centric system is to design for the layman users. So, we need to know the users. Now, contextual inquiry is one way to know the users based on that we get some idea. Along with that, it may not be possible to get all information for a particular user group through a contextual inquiry or one inquiry, you may need to go for a lengthy one or multiple times to get a complete picture. But sometimes we can avoid that delay or the effort. In fact, sometimes that may not be feasible at all. So, we can avoid that by taking help of design guidelines.

So, they are basically a way to give you a starting point that based on this, we think of a design as a starting point and then, keep on refining it based on the iterative cycle that we have seen in the development lifecycle stages.

(Refer Slide Time: 02:58)

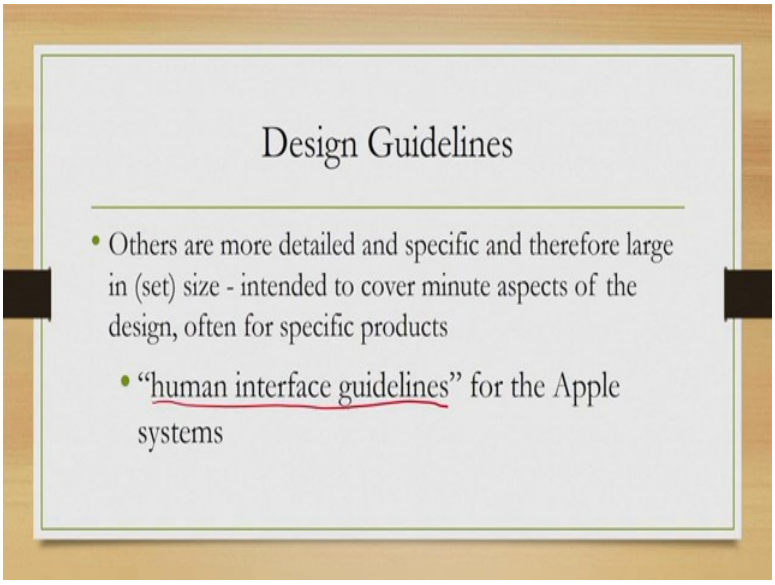
Now, there are many design guidelines which is expected.

(Refer Slide Time: 03:01)

Some are very generic ones. Now, this generally guidelines have the characteristics that they are small in size; that means, the number of guidelines are actually very small because they are generic. So, they basically cover the broad aspects of interactive systems at a rather high level.

A couple of examples are the “eight golden rules” of Schneiderman proposed in 1986 or the “seven principles” proposed by Don Norman in 1988. There are many more such generic set of guidelines of course, but these two are quite popular or were quite popular.

(Refer Slide Time: 03:35)



There are also very detailed set of guidelines, where the number of guidelines are really very large and these guidelines are intended for design of minute aspects of a particular product.

So, typically they are designed for specific products or specific brands, for example, the “human interface guidelines” which is used by the Apple products. So, these guidelines was proposed for development of the Apple systems. So, this discussion on the detailed guidelines is not a good idea because they are specifically designed for specific products.

Instead what we should do to understand the idea of guidelines is to discuss one set of broad guidelines, generic guidelines which will be helpful in giving an idea of what are the guidelines; what purpose they serve and how they can be utilized in the design of interactive systems.

(Refer Slide Time: 04:33)

So, we will do that in terms of the Eight Golden Rules proposed by Ben Schneiderman in 1986. As the name suggests there are 8 rules or a set of 8 guidelines. The first one is called strive for consistency; what it means? So, this means that our design whatever we are designing should be following consistency in its approach.

Now, consistency can be thought of at two levels; the internal consistency and the external consistency. Now, internal consistency is that inside the system whatever you do the way you design should follow a consistent approach. For example, suppose you are designing an interface where there are multiple stages, multiple gui's through which a task can be performed.

Now, in one interface, you provided the close option using a red button. So, if a user clicks on the button, then the interface closes and the next interface appears. In another interface, you have done the same thing using a green button. So, now, if the user clicks the green buttons, then the interface closes and another interface appears. So, in one interface you have the red button to indicate closer of the interface, in another interface you have a green button to indicate closer of the interface.

This of course, as you can understand violates the internal consistency. Different parts of the same system have different design decisions, which is violative of internal consistency. So, internal consistency means that throughout the system, we should follow the same set of principles to design the system.

Now, external consistency is somewhat different compared to internal consistency. Now, when we see around us we get to see certain conventions, certain behavioral traits, certain patterns in the design. We need to actually implement those patterns behaviors or convinces. Now, if we violate from there, then that is actually a violative of external consistency.

Again, let me give you some example. So, typically when we talk off closer or stoppage or ending something, we see that all around us we get to see the use of the red color. For example, in the traffic light you may get to see this red light to indicate stoppage of the car or in some electrical switches you may get to see the red buttons to indicate closer of the circuit.

Now, suppose you are designing an interface where instead of red you are using the color orange to indicate closer, then definitely this is a violation of our conventions. So, what we get to see around us and what we are seeing on the interface are different. So, our objective should be to avoid that type of violations; otherwise the naturalness or the way user perceives the interface may get affected.

So, if the user gets to you see that whatever he or she sees around him or her preserve is what, he or she has to learn in order to use an interface are completely different. Then, definitely that creates anxiety, that may affect performance, that may increase the number of errors and many more problems may come up. So, we should try to avoid violation of external consistency and also violation of internal consistence.

(Refer Slide Time: 08:31)

The second golden rule is the rule for universal usability. So, we should design for universal usability; what it means? It means that our design should cater to all possible user groups. Now, recollect from our earlier lectures that we can have a broad categorization of users in three groups; novice, expert and intermittent.

So, typically when you design something we should have facilities for all these three groups; novice, intermittent and experts. For example, as we have already seen that in a word processor say Microsoft word or WordPad. We typically have two ways to close or save a file; one is through menu and one is through the use of a combination of keys or hotkeys say control plus S; if we press these two keys together as a combination, then the file gets saved. Alternatively, we can save a file by using the menu option file; under file sub menu option save and we can save the file.

Now, the menu-based file saving is designed for novice user's, hotkey-based file saving is designed for expert users. So, this design tries to take into account the universal usability concern.

(Refer Slide Time: 09:52)

The third golden rule is that offer informative feedback. So, whatever the user is doing the user should get a feeling of how he is performing. So, for example, suppose the user wants to save a file and the user presses the file menu option under that the save sub menu option; but then, nothing happens. There is no visible change on the interface.

So, then the user will not be sure whether the file has been actually saved or not instead if you provide some change in visual appearance of the screen, then probably the user will be more comfortable. For example, you may show that there is a progress bar which is showing the saving of the file and once the file saving is complete, the progress bar exits.

So, user gets a feedback that the file is now getting saved and it is saved since the progress bar has disappeared. Similarly, in menu-based option if there is a floppy menu icon, so before saving the icon was active and brighter after saving if the color of the icon changes to a very relatively dull color; that means, the save took place. So, this relative change in color may indicate that something happened and the user may become less anxious otherwise without any proper feedback, the user may not be feeling very comfortable with the system. So, giving feedback is one principle that we should follow.

(Refer Slide Time: 11:19)

Now, the fourth principle or golden rule is basically design dialogues to yield closer. Now, when you are performing some task on your interface and if the task end suddenly, then you may feel like you are out of control, the system is behaving in a very unpredictable way. It is giving you surprises which may lead to your anxiety, your worry and you may not like to use the system again.

Instead if there is a sequence, if there is a systematic way of letting you feel that you are in control the system, you are actually doing the things you intended to do; that means, everything happens gradually and with proper feedback to you as a user, then actually it helps you or the it helps the user feel in control and that is what this golden rule says designed dialogues to yield closer.

So, essentially there should be a dialogue to which should be properly designed to indicate to you that you are actually closing an interaction that you have started. For example, suppose you are planning to perform an online purchase. Now, there are several sub stages in that and you need to perform several sub task like putting the search string to load the purchase website, selection of items, choice of an item, putting the items into the cart, checking the carts, making online payment, inserting your delivery address and so on many many activities.

Now, if this interface is not properly designed, then you may feel like the steps are not being done properly and you may feel worried. Instead suppose let us organize these

activities into three broad groups; start of the activity or beginning, middle and end. So, at the beginning you input the search strings and watch the things to being loaded; middle stage involves you to select the items and put them into the cart and at the end you may like to check out and make the payment.

After each stage if you get to see some feedback like say for example, after you put something in the cart, there is a cart symbol appearing with the number of items in it displayed, then those feedbacks give you some sort of confidence that ok, I have put it in the cart and it is actually getting registered by the system. Then, definitely that helps you progress to the end stage, otherwise you will be wondering whether the cart items are actually there; whether your selections are registered or not; whether you re select those. So, many such issues will come up.

So, organizing various sub tasks into groups providing feedback at the end of each sub task and organizing all these things in a way of a dialogue between user and system helps you or the user feel in control of the overall system which in turn increases the usability. So, this is the fourth rule.

(Refer Slide Time: 14:27)

Next comes the rule of error prevention and handling, two r is human that we all know.

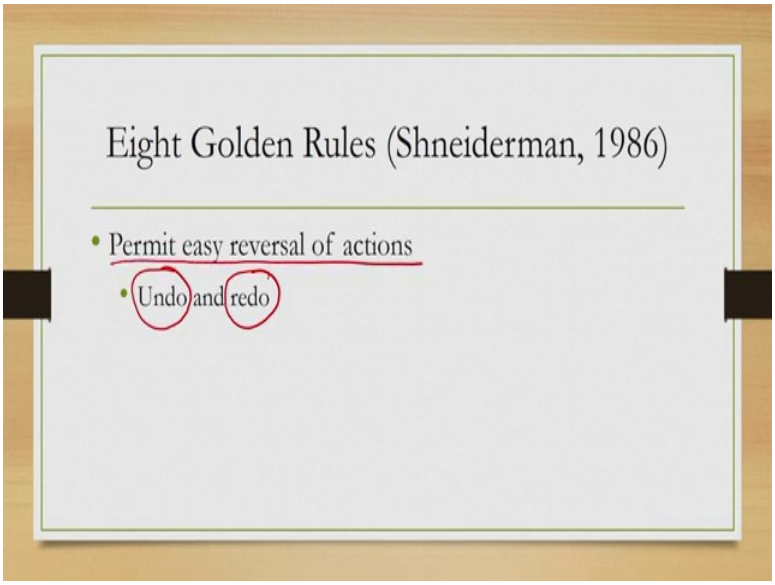
So, there is likely to be errors and you may not be able to avoid errors completely. However, what a designer can do is design things in a way such that the chances of

errors are reduced to the maximum extent possible. Some very simple things can be followed. A very simple thing to do is basically do not keep the start and close buttons close by, then there is a possibility that by mistake somebody is closing the application occasionally.

Another important thing is in order to handle errors; you should give proper error messages and probably instructions to the users to come out of the error stage. So, if you saw a complicated error message, if you recollect in one of the earlier lectures, we have seen an error message were showing to the user some error code, some service name application security manager and so on. These are actually not very informative to a layman user because the user does not understand what is this hexadecimal code means; what the service does and what to do.

So, instead the messages should be in the natural language as much as possible, avoiding technical terms and be precise and actually indicate to the user what to do in the event of an error. These are very important considerations in the design of any interactive systems. Related to this is another golden rule that is permit easy reversal of actions.

(Refer Slide Time: 16:07)



So, essentially it indicates the availability of the features such as undo and redo. I think most of you are already aware of these features and probably all of you are or most of you are actually doing these things. So, what purpose these undo and redo features serve? When you make an error, the first thing is that you are worried whether the

system is going to crash, what will happen to my data and so on. Now, if there is some undo or redo facility, then you know that even if you will make some error you have there is always a possibility to come back from where you started.

So, at least you can reduce the damage, minimize the damage and you can restart whatever you were doing. But if there is no such facility, then you will always be worried and anxious, whether you will you can never get back what you are doing and that anxiety may lead to abandoning of the system. So, it is always a good idea to have facilities like undo, redo which is formally called permitting easy reversal of actions.

(Refer Slide Time: 17:19)

So, you want to reverse some actions that you have erroneously done. The seventh rule of course, all the rules are related to each other. The seventh rule states that keep users in control; so, here what we need to do is basically we design in a way such that the user feels that they are in control.

So, if we are if the system behaves in a very unpredictable way like you clicked on something, then nothing happened visibly, but in actually inside the system something happened some. So, then that is very unpredictable behavior for the user. The user does not get to know what is happening, some surprising actions.

All these things actually make the user feel not in control, which in turn lead to anxiety and worry which in turn may lead to dissatisfaction and abandonment of the system. A

very good example is the command line based interfaces disservices, drag and drop base interface. So, in command line based interfaces, if you put a command then, you do not know how the things happened and the result is produced.

So, essentially you are in the dark, how the result is produced in drag and drop type of activities. You are actually supposed you want to copy a file, you are actually dragging the file, you are selecting the file and dragging it to the destination folder and it gets copied, you get to perceive what is happening and you will get a feel that ok, you are actually really copying the file rather than you are relying on what the system is telling you.

That type of feeling, that type of perception actually helps in letting the user feel in control which in turn helps in increasing the usability of the system.

(Refer Slide Time: 18:57)

The final golden rule is a very interesting one, it is called short-term memory load. This is one of our physiological constraints that has been captured in the form of a rule and used successfully in the design of interactive systems. The rule was actually proposed by George A Miller, the cognitive psychologist long ago in 1956, but the rule has found very wide application in the area of interactive system design. It is also known as Miller's rule or "7 plus minus 2" rule.

What it states that our memory actually contains broadly two components; long-term memory and short-term memory. In the short-term memory at a time, we can hold at most we can hold between 5 to 9 chunks of information's, 7 plus minus 2; 7 plus 2, 9; 7 minus 2, 5; between 5 to 9 chunks of information. So, at most nine chunks of information were a chunk of information is unit.

So, it can represent a character, it can represent string of characters, it can represent a symbol anything, but essentially here it refers to a unit of information. So, in other words, like our computer RAM the short-term memory acts like a scratchpad and volatile memory which at a time holds a limited amount of information which gets erased when the new task has starts.

And so, the implication to the designer of an interactive system is that they should not force the user to remember too many things which goes beyond the capacity of our short-term memory. For example, if you are asked to remember a series of more than 9 commands to perform an activity with so, you need to recall it from the main memory to short-term memory.

Now, as I said short-term memory can hold maximum 9 units of information. So, the moment you need to recall more than 9 units of information, you may not be able to recall it properly and which may lead you to make errors to do things that you would never intended to do and so on. So, essentially this rule 7 plus minus 2 rule or the Miller's rule tells you how to make a design that takes into account this human physiological constraint of maximum 9 chunks of information in the short term memory during execution of a task.

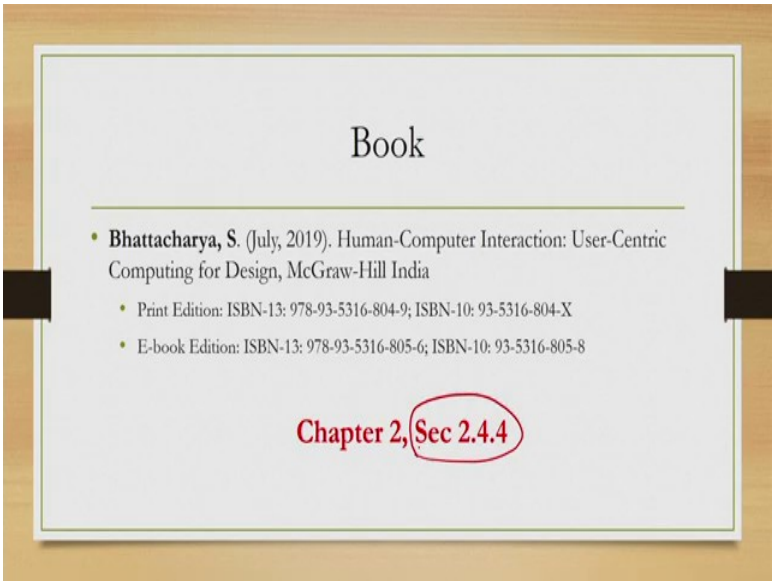
So, together the set of eight rules provide us a way to think of a design at the beginning. Of course, these rules do not tell us exactly how to design, but some broad idea what we should avoid, what we should keep so that these rules are not violated; some broad idea and that is the idea of any guideline just to recollect. So, there are eight rules.

Now, these eight rules are essentially meant to give you some starting point in design; first is strive for consistency, this is one rule, rule number 1. Then, second is design for universalizability, rule number 2. Third is offer informative feedback, rule number 3. Forth is design dialogues to yield closer that is rule number 4. The fifth rule is related to

error prevention and error handling. So, offer error prevention and simple error handling facilities in your design this is the rule number 6 rule number 5.

Rule number six is permit easy reversal of actions relate to undo and redo facilities. The next rule is keep users in control and the final and one of the very interesting rule is reduce short term memory load which came from the Miller's rule which states that at the time we can hold maximum 9 units of information in our short term memory. So, any design that we do should not force us to recall more than this amount or should not force us to remember more than nine units of information.

(Refer Slide Time: 23:14)



So, the material I have covered today in this lecture is taken from this book; chapter 2 section 2.4.4. So, all the things you can find in this section of the chapter 2.

Thank you and goodbye.