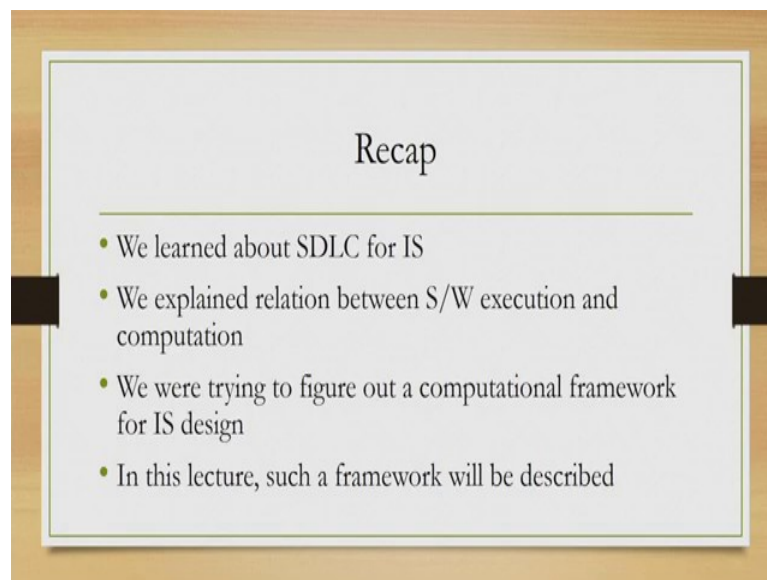


User-Centric Computing for Human-Computer Interaction
Prof. Samit Bhattacharya
Department of Computer Science & Engineering
Indian Institute of Technology, Guwahati

Lecture - 10
The UCC Framework with Illustrative Case Study

Hello and welcome to the course User-Centric computing for Human Computer Interaction. Today, it will be the 10th lecture, before we go into the content of today's lecture. As usual we will first recap what we have learned in the previous lectures, particularly in the last week.

(Refer Slide Time: 00:55)



So, in the last week we started with a description of the interactive system development lifecycle. So, essentially that is a software development lifecycle tailored for the development of interactive system. We required to know this lifecycle in order to have a structured view of the different stages or different processes involved in the development of an overall interactive system software.

Now, along with that we briefly introduced the concept of computation in the context of interactive system design. So, essentially that is another way of understanding at a more deeper level the development process.

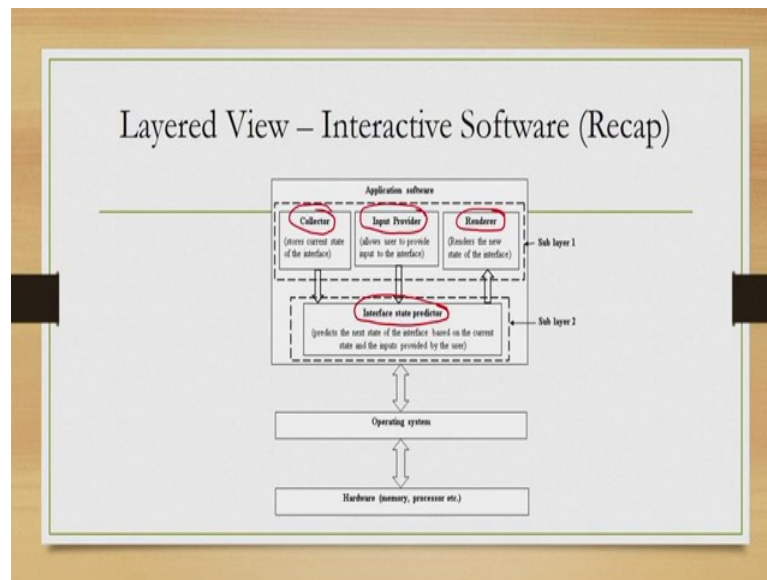
So, when we are talking of the design of an interactive system we mostly rely on intuition and experience of the designer which is somewhat informal. But, instead if we try to understand the computation involved in the software that is the end product of the design then we may be able to have a better understanding of how to design that software. In that context, we discussed the relationship between a software execution and a computation in a traditional sense.

So, just to recap the computation is essentially change of state which we have discussed and software execution also can be mapped to the change of state in a discrete state machine and can be considered equivalent to the traditional notion of computation.

Now, when we are talking of software execution as a computation. So, what we want to achieve is basically a structured way of understanding this computation and for that one way to do that is basically to view a software as a collection of layers. Now, these layered view helps developers have a better understanding of the overall software and, this layered view can aid the designer of the final interactive software product as we will see after today's discussion.

We started with a very simple three layered view of software where at the top is the application layer, in between is the operating system layer and at the end is the hardware layer. And we discussed that it is not necessary for an application program developer to know in details about the operating system or the hardware or how they interface, all these minute details. Instead, the designer of an application program can concentrate on the design of the program itself and rely on some well defined standardized APIs or system calls to interact with the hardware resources through the operating system.

(Refer Slide Time: 04:03)



From there we learned about a slightly modified view of the application layer where we talked of two sub layers. As you can see here in this figure in the sub layer one we have three programs; the collector program, the input provider program and the renderer program, and in sub layer two you have one program that is interface state predictor.

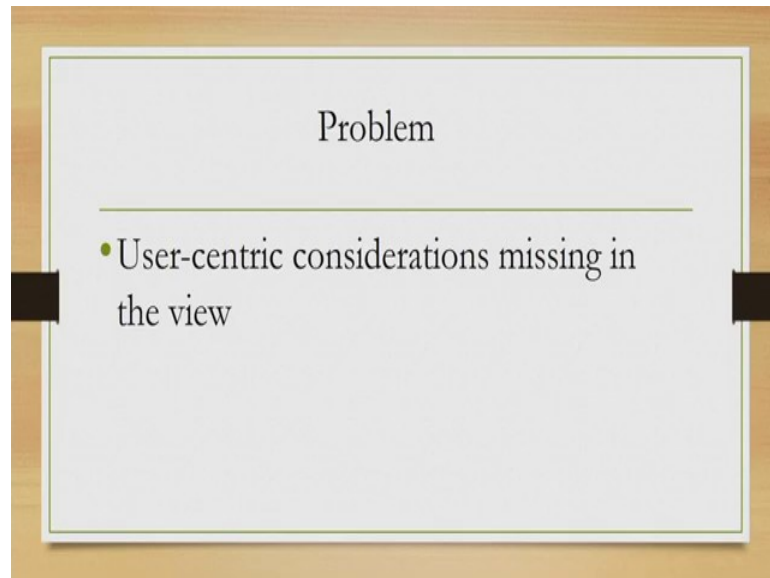
So, the purpose of these programs or the way to view this application software layer as a composition of two sub layers with the individual components in it, is basically to represent the way a software developer or the developer of an interactive system software is likely to view an interactive system software.

So, as per that view there are four components four programs that are part of the software. The first program is one input provider through this program the user can interact with the software and user can provide input, the job of this software or this program is also to capture that input. So, that other stages can make use of the inputs that are coming from the user.

Once this input is there, there is likely to be another program which basically captures the state of the interface and then based on this state information and the input provided by the user a state change happens. And, this state change is captured by a third program interface state predictor which predicts the next state of the system of the interface.

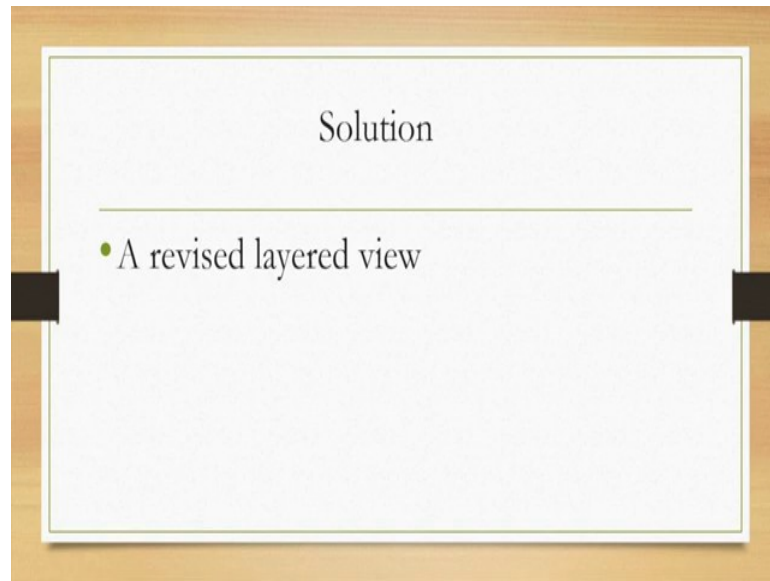
And from there this next state information is passed on to another program which actually renders the state on the interface, for the user to perceive to see to hear essentially to perceive the change. So, that is a simplified view of the software organized into two sub layers with one layer having three components and other layer having a single component.

(Refer Slide Time: 06:41)



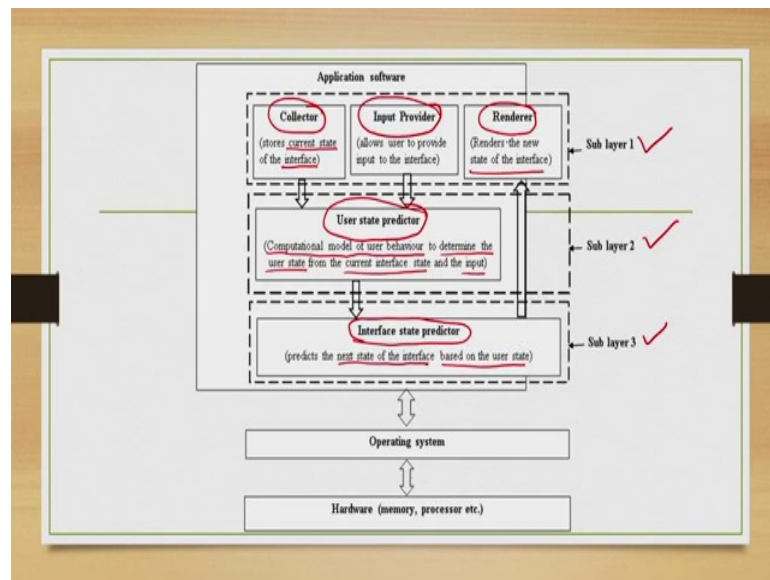
But, the problem in this view as we have discussed in the last lecture is that it does not consider anything user centric explicitly. In other words, whatever programs we are talking of whatever components we are talking off do not explicitly deal with the user characteristics, the user behavior, the user centric considerations. So, what we need is essentially a modified view of the layered architecture.

(Refer Slide Time: 07:08)



So, that is what we are calling a framework to understand the user centric computation in the interactive system software.

(Refer Slide Time: 07:17)



So, let us have a look at this modified layered view which we are calling as a framework for understanding how user centric software ideally should work. So, here as you can see instead of two there are now three sub layers 1 2 and 3; earlier there were two sub layers, now we have three sub layers. Sub layer 1 remains the same it consists of an input provider component, renderer component and a collector component. The input provider

component again the meaning the semantics remains the same, it is essentially a program that allows interaction that allows the user to provide input to the system and captures that input.

The collector it is meaning also remains the same with respect to our earlier layered view where the job of the collector is basically to store the current state of the interface. And, the job of the render is to render a new state of the interface which will be predicted by the interface state predictor.

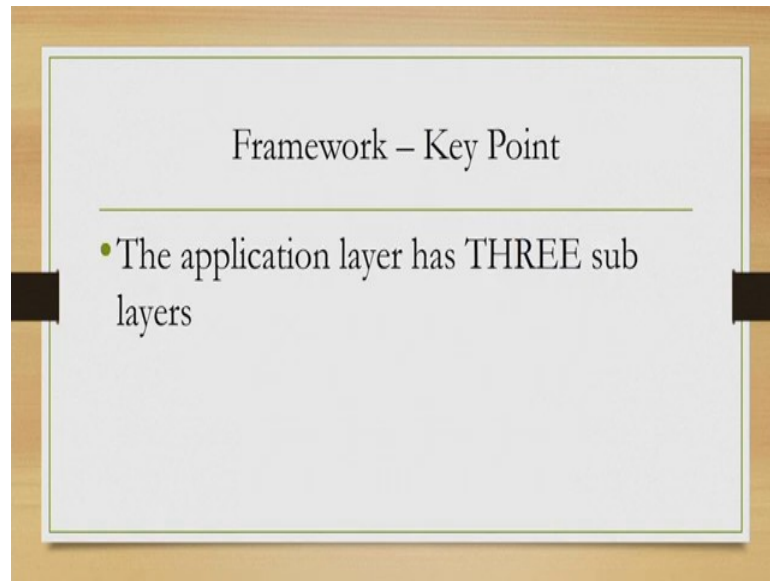
So, essentially layer one with its components remains the same as in our previous discussion. Sub layer 3 now contains interface that predicted, earlier it was part of sub layer 2 and the meaning of the interface that predictor is the same; essentially, takes as input the current state and predicts the next state.

However, here the nature of the input to this module is changed. So, it is not only the current state of the interface, but along with that it takes as input the user state. And, we have introduced a new sub layer that is the user state predictor; now, that as the name suggests it is an explicit incorporation of the user centric concerns that we face in the design of interactive system.

So, what this user state predictor does? It is essentially a computational model of user behavior which is used to determine the user state from the current interface state and the input. So, earlier this current interface state was being passed on to the interface state predictor to produce the next state which was sent to the render for display.

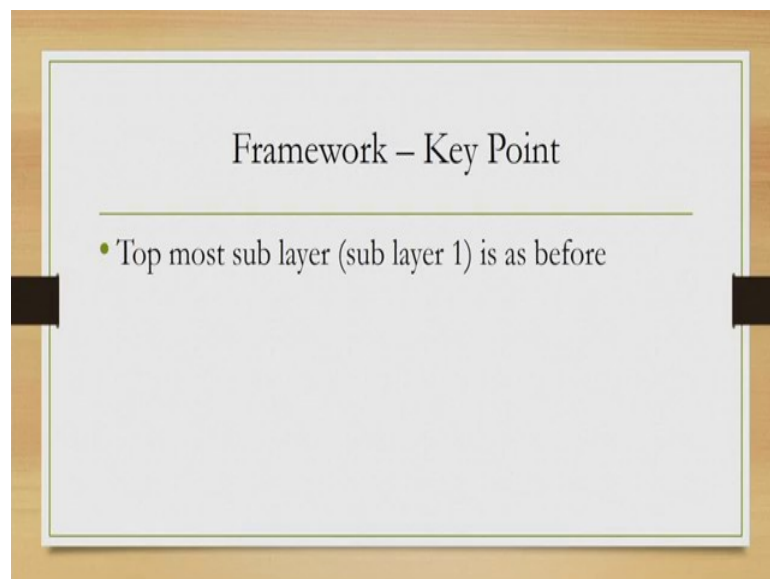
Now, this current state captured by the collector module and the input captured by the input provider module or program are not sent to this interface that predictor instead these are sent to the user state predictors. So, based on this current state and the input this module computes next state of the user of the system, and that next that information is used as input to the interface state predictor which predicts the likely interface conforming to the next day and that state is that interface that is now passed to the renderer. So, there is a significant modification done.

(Refer Slide Time: 10:39)



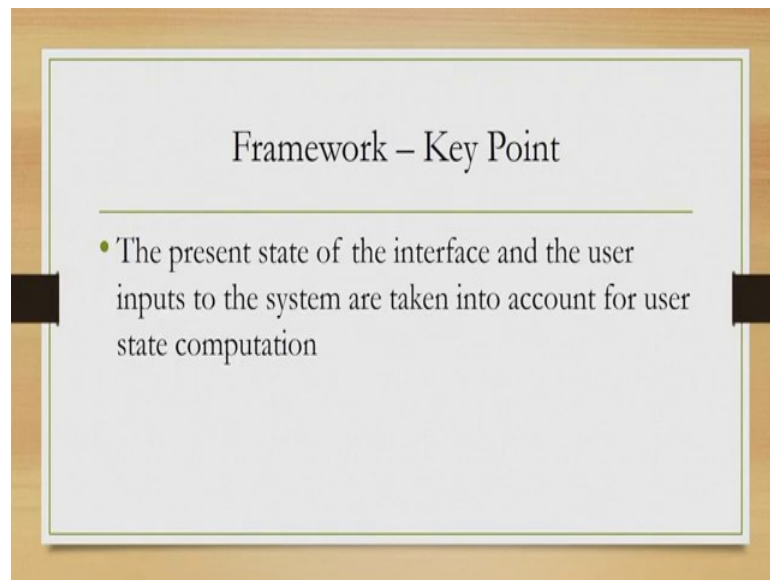
So, now the application layer has three sub layers as we have seen.

(Refer Slide Time: 10:44)



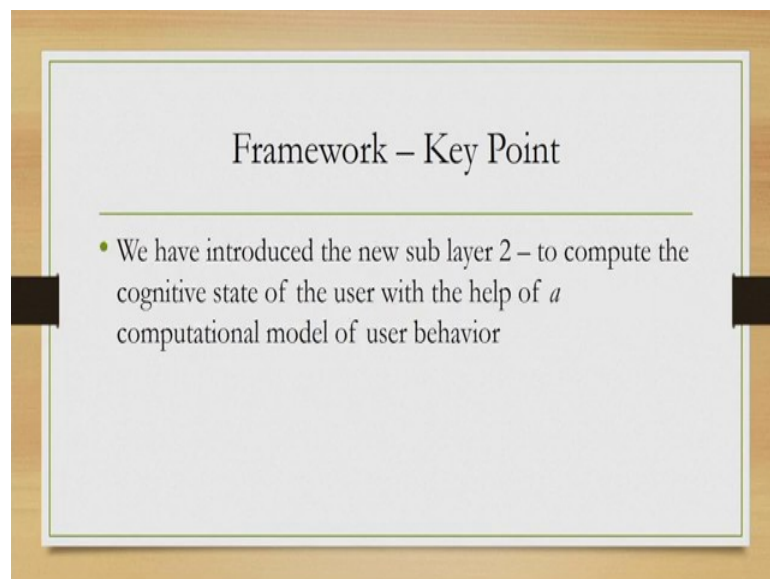
And the top most top layer remains the same.

(Refer Slide Time: 10:49)



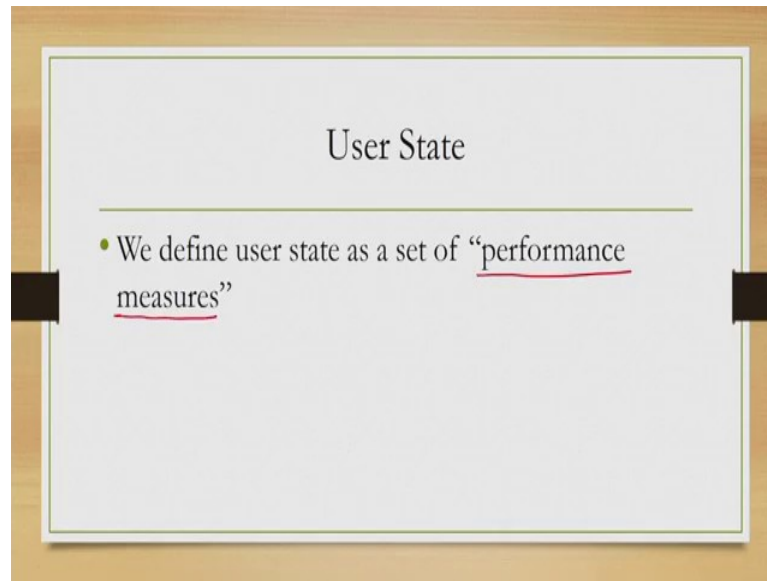
Along with the sub layer 3 which was earlier sub layer 2.

(Refer Slide Time: 10:53)



But sub layer 2 now in the new system which is the user state predictor has been introduced which explicitly takes into account the user centric concerns that we are likely to face in the design of an interactive system. Now, let us try to understand this new sub layer that is the user state predictor.

(Refer Slide Time: 11:19)



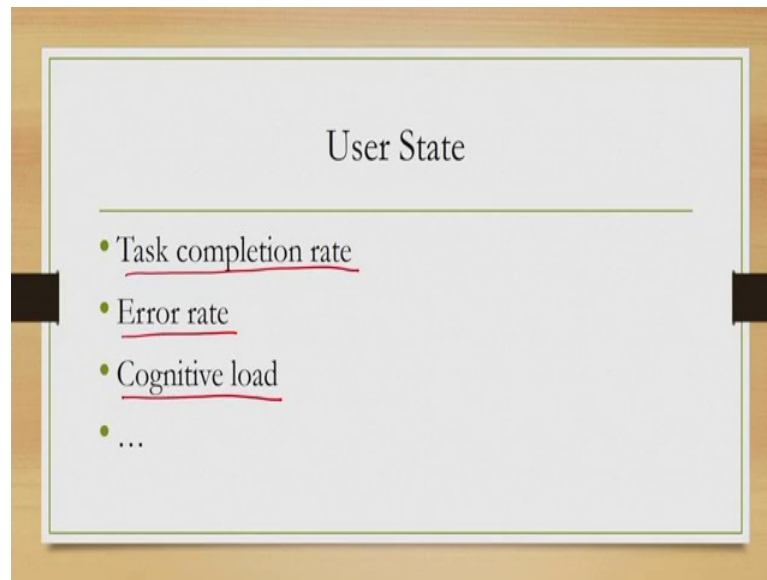
So, this is clearly one very important component and this is going to be the most important component for interactive system design because it explicitly takes into account the user concerns which ultimately is our objective to have a usable system. Now, remember that when we were talking of a state in the context of computation, we referred to a set of variables, we usually define a state as a set of variables. So, when we assign a set of values to these variables when we instantiate these variables with values then we get a particular state.

So, the definition contains variables and realization of the state is nothing, but assignment of values to these variables. Now, when we are talking of user state again if you can recollect we talked of user state as a state of mind or state of cognition a current state of the process of cognition, but how do we define a state, state of mind or state of cognition.

So, we have already noted before that we still do not have complete knowledge of human cognition process, we still do not know what defines, what regulates or what controls human cognition. So, when we are talking of cognition as a change of state from some initial state to the goal state, it is not possible to capture the state in terms of all possible variables that actually control cognition because we ourselves do not know. Instead, what we can do is basically come up with an approximate definition that suits our purpose.

So, in the context of user centric computing that approximate definition of a user state can be obtained in terms of performance measures. So, this performance measures are essentially an indirect representation of the user state, what are these performance measures?

(Refer Slide Time: 13:28)



Some examples can be task completion rate, error rate, cognitive load and many more such measures we can define. As you can see here, these measures are not directly related to what happens in our mind, instead they are measuring some outcome of the process of cognition, but with this measures we can actually capture the progression of cognitive process and the change of state.

In other words we can use these measures as an indirect representation of the state of cognition and these measures provide us the set of variables we need to define a cognitive state.

(Refer Slide Time: 14:18)

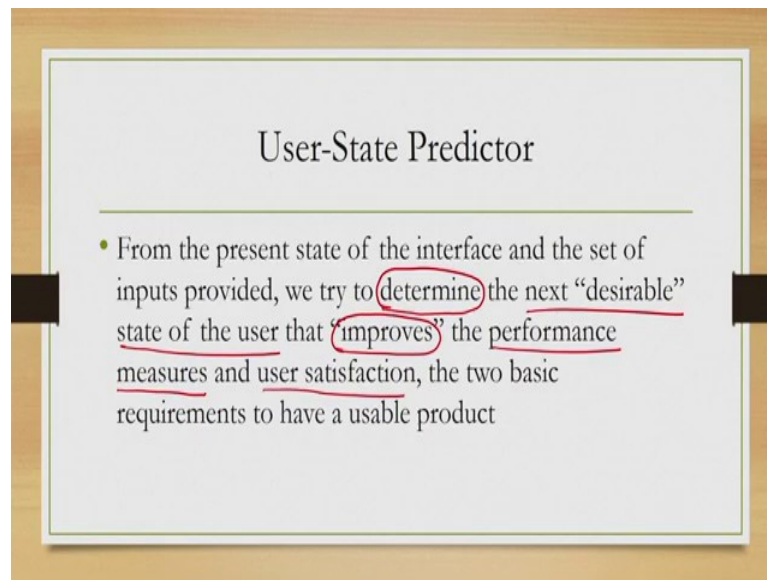
User State

- At any given time, we compute the values for these measures
- Set of computed values represent the state at that instant

So, what we do with this measures so, at any given time we compute. So, note this term compute we compute the values for these measures. So, once we computed those values then the set of computed values represents. The cognitive state of the user at that instant, it is an approximate representation. I am again emphasizing on the fact that it is not exact representation of cognition because we ourselves do not know what is exactly the exact representation.

So, we are dealing with approximate representations and one way of representing cognition approximately is in terms of set of performance measures. So, we compute their values at an instant and that the set of those values indicate the state of cognition of the user at that instant. So, once we computed the state by computing the set of performance measures what next?

(Refer Slide Time: 15:32)



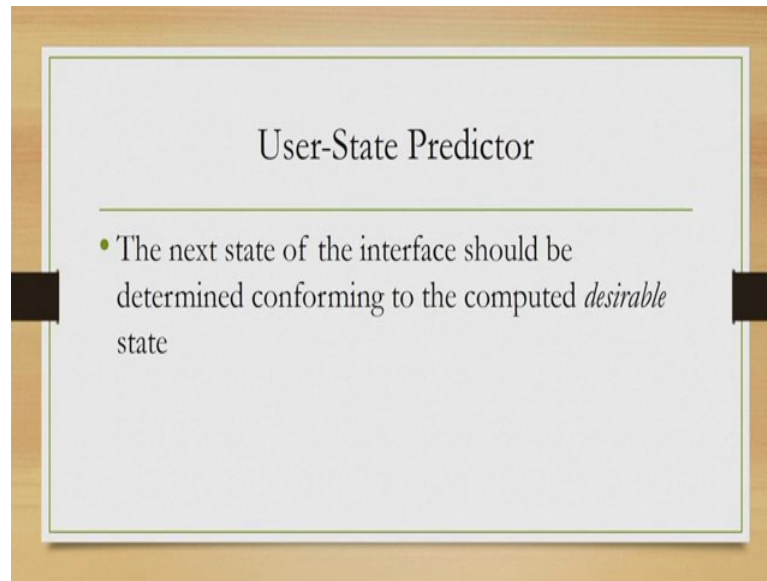
User-State Predictor

- From the present state of the interface and the set of inputs provided, we try to determine the next “desirable” state of the user that “improves” the performance measures and user satisfaction, the two basic requirements to have a usable product

So, next what we do is basically from this current state we try to predict or determine the next desirable state of the user. Now, what is a desirable state? Clearly, a desirable state is the one which improves the performance measures and user satisfaction. Now, both of these are part of the definition of usability if you may recall from our earlier discussion. So, when we manage to improve both, then definitely we are going to have a state which is usable and ultimately it will lead to an usable product.

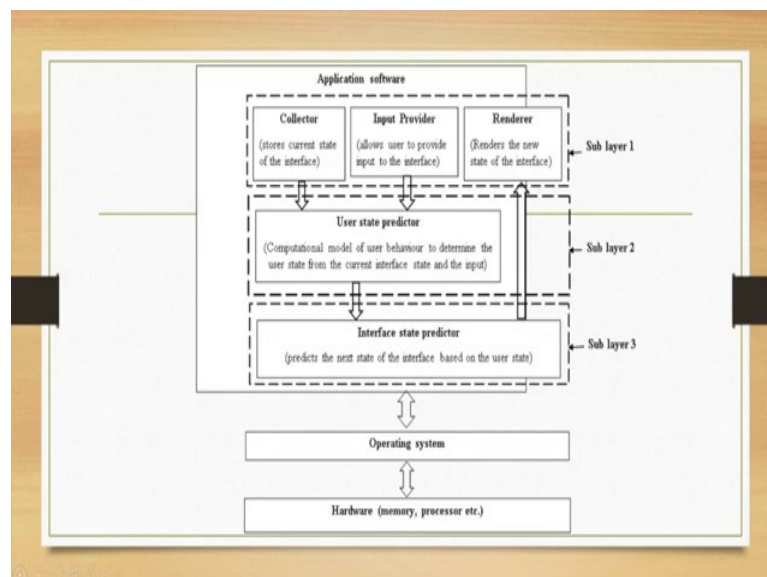
So, just to repeat again, once we managed to capture the state in terms of performance measures in terms of computing the performance measures what we do is basically, we try to predict or determine the next desirable state by that what we mean is a state that improves by some means, the performance measure and user satisfaction which are fundamental concerns in developing a usable product.

(Refer Slide Time: 17:02)



And, once we managed to get the next desirable user state then we pass on this information to the interface state predictor. So, the next interface state should be such that it conforms to the desirable state. So, all these at this stage may seem slightly abstract to you, but soon we will have one example to illustrate these concepts with more clarity.

(Refer Slide Time: 17:28)



So, then what is the purpose that this framework starts? So, to recollect in our framework we have three layers. Layer 1 contains three sub components those are the collector

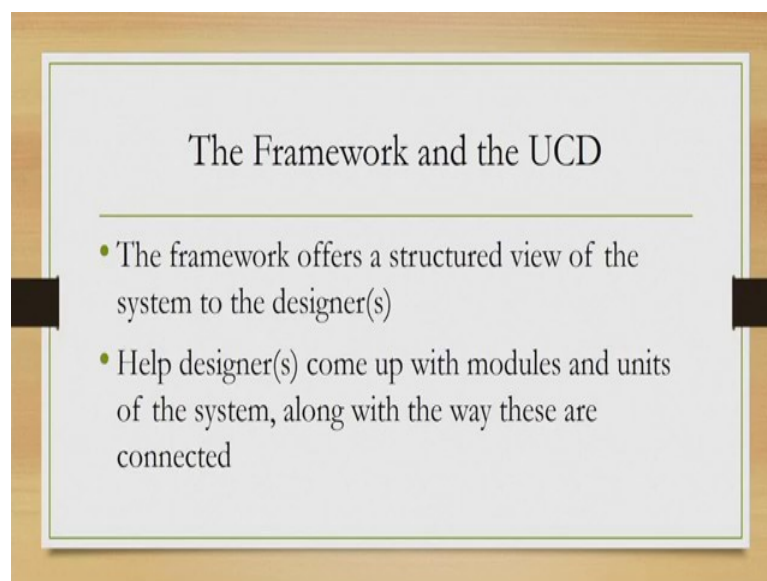
which collects the current state of the interface, information related to that then we have input provider which allows the user to interact with the system and finally, we have the renderer which actually responsible for rendering the interface. So, that the users can perceive it either through vision or other senses including auditory senses.

In the sub layer two, we have a very crucial component which is called user state predictor, it takes as input the current state information of the interface and the inputs that the user provides. Based on that it computes where the term computation should be kept in mind, it computes the next desirable state of the user.

Now, that next desirable state is defined in terms of improvement in the variables that defines the user state. So, here we are using an approximate state definition where we represent a user state in terms of a set of performance measures such as task completion time, error rate, cognitive load and so on.

So, the next desirable state is the one where these measures are improved the performance is improved. Now, once that state is computed what well do is pass it on to sub layer three which is the interface state predictor, which based on the computed state produces specification of an interface that conforms to this improved state or desirable state, and this specification is passed on to the renderer program which renders that perception of the user.

(Refer Slide Time: 19:43)



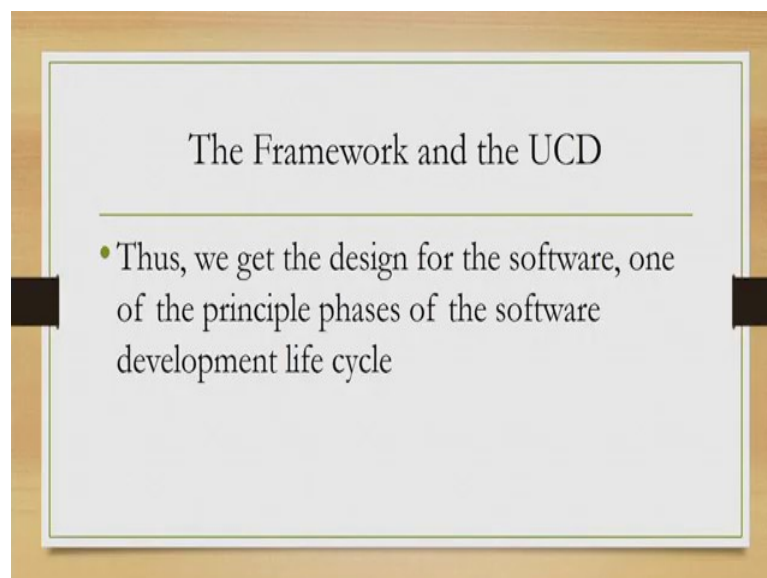
Now, with this understanding of the overall computation in user centric software, what do we get? We get of course, a more structured and formal view of the system. How it helps the designer? So, now, the designer can think of the design as consisting of various components.

For example, the designer may think of the design as consisting of at the very basic level five components the collector, input provider, renderer, user state predictor and interface state predictor and concentrate on designing each of these components.

The designer may also think of the final software as a competition of many more programs than this basic five. For example, the interface that predictor may further be subdivided into multiple programs, the user state predictor may further be subdivided into multiple programs, the input collector may have multiple programs. So, each of these programs we call in more formal terms that modules sub modules.

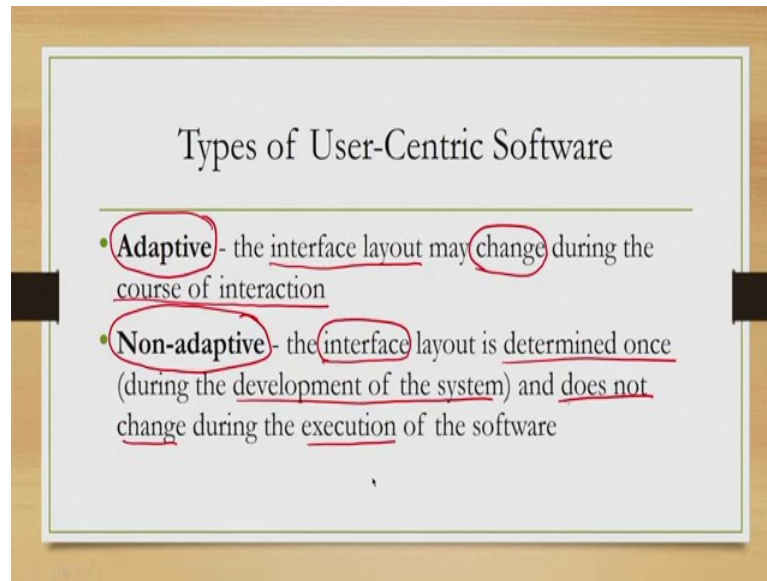
So, it gives an idea to the programmer to the developer how to think about an interactive system and that is the main purpose. So, once this idea is there, the designer can come up with better designs which can actually help in visualizing or understanding the overall systems.

(Refer Slide Time: 21:28)



So, that is the main motivation for coming up with a framework, basically to help designer have a more formal approach to the overall design of the system.

(Refer Slide Time: 21:41)



So, once we know how to organize one interactive software in the form of modules and sub modules, the next question comes is do it all these components in all the systems. That is a very important question because in reality there are many types of interactive systems; broadly, we can divide them into two groups adaptive and non-adaptive.

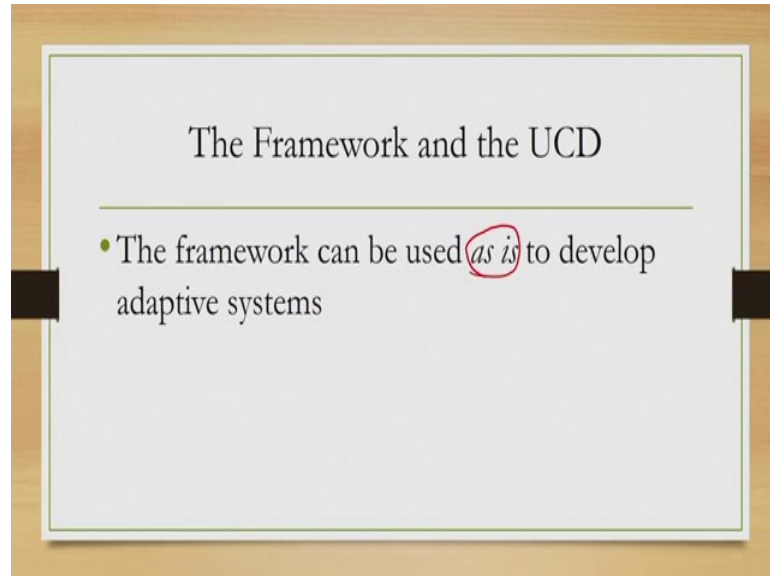
Now, we call some system adaptive when the interface appearance the layout, the interaction may change during the course of interaction. So, in other words you are interacting with the system and what is being rendered at this moment may not remain same in the next instant depending on your interaction it may change; similarly, the way we interact may also change. So, adaptive means the system adapts to the user behavior by changing the appearance of the interface or changing the interaction.

In contrast, when we are talking of non adaptive systems, what we refer to is basically the interface remains static, the interaction mechanism remains static. So, there is no change. So, what we determine the interface once and it does not change during the execution of the software. So, the interface is designed during the development time and does not change during the execution.

In adaptive, it may there may be one designed during the development at the end of the development cycle and that design may keep on changing during execution. In non adaptive that is not the case whatever we get at the end of the development cycle,

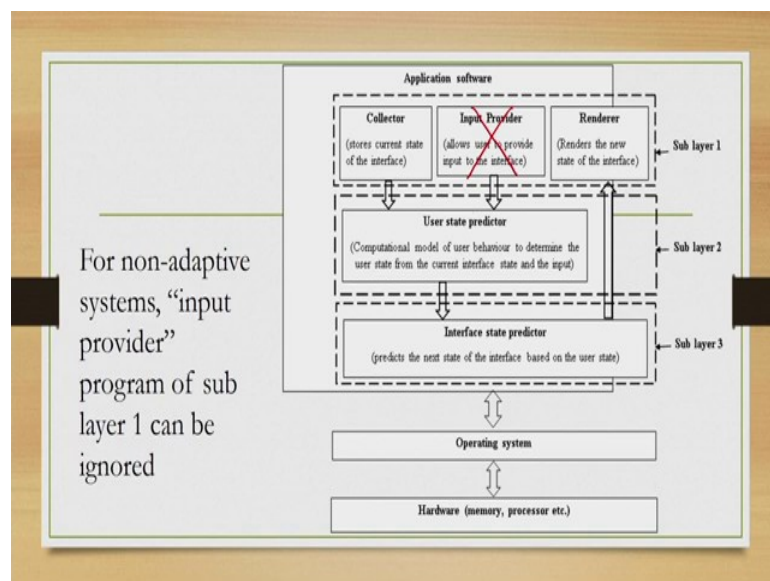
remains same during the execution of the software. How do we relate our framework to these two types of systems? In case of adaptive system we use the framework as is.

(Refer Slide Time: 24:23)



So, when we are talking about building an adaptive system, the framework remains whatever we have discussed, all components will be used and all components are relevant to understand an adaptive system.

(Refer Slide Time: 24:44)

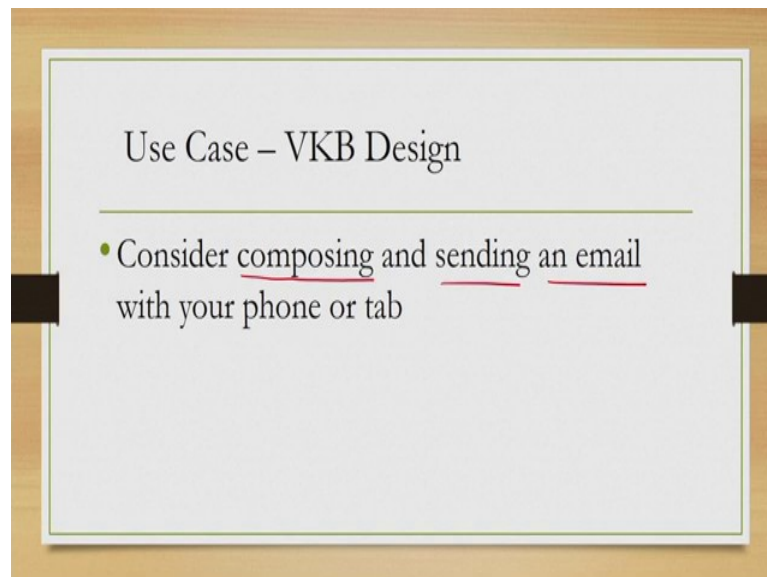


In comparison when we are talking of non-adaptive system we may not keep the framework same. So, here we can ignore this input provider because here nothing

changes based on the input. So, during the development stage whatever is developed remains the same; so, nothing varies based on input so, nothing changes during execution. So, input provider have no relevance in understanding and non-adaptive interactive system.

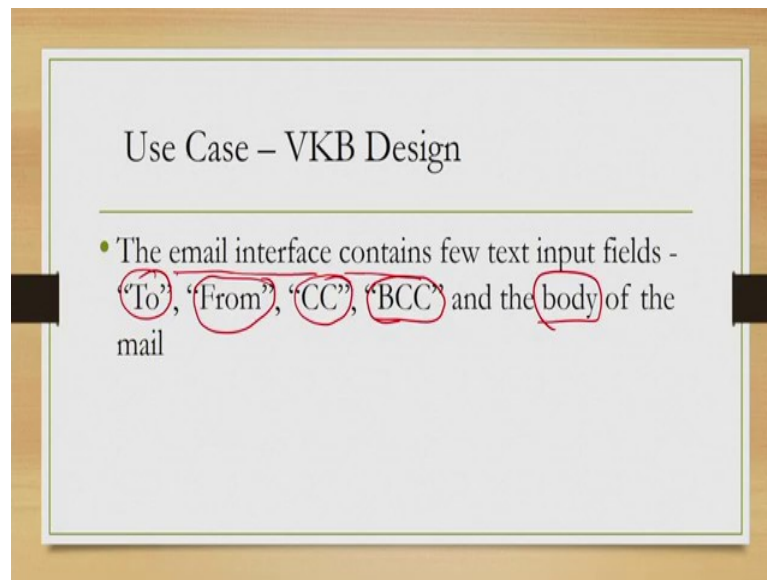
So far whatever we have discussed may seem somewhat abstract the idea of user state, the idea of systems adaptive non-adaptive, the idea of these layers of the framework. Let us try to understand this framework with a case study that will better illustrate the framework.

(Refer Slide Time: 25:47)



So, what is our case study? The case in question is composition and sending of an email using your phone or tab. So, we want to basically create and send an email with your phone or tab. What you are likely to do, if you are asked to do this. Of course, all of us do it very frequently and we probably never notice what are the things that we are doing.

(Refer Slide Time: 26:15)

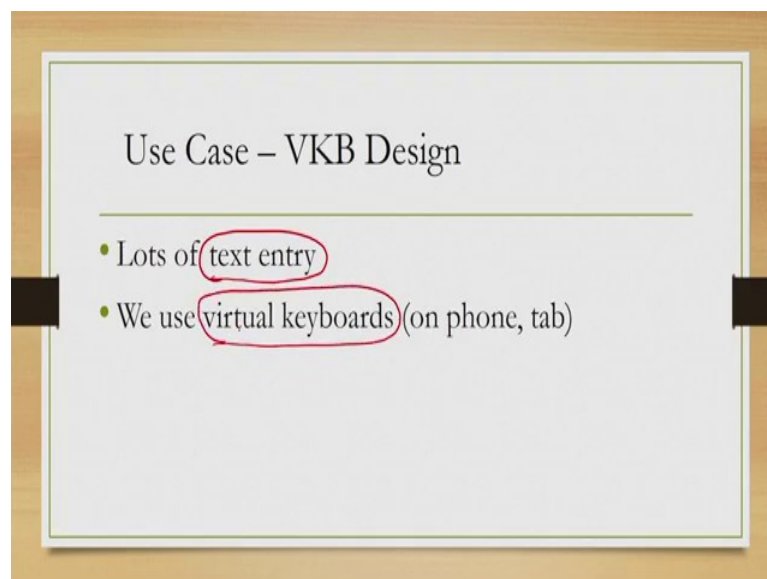


Use Case – VKB Design

- The email interface contains few text input fields - "To", "From", "CC", "BCC" and the body of the mail

Now, the first thing we should note is that the interface that is there that you get to see for mail composition on your phone or tab, this email interface contains few text input fields those are this two field. So, the sender; the address of the receiver, the address of the sender or the identity of the sender, identity of the receiver in terms of mail IDs. If there is any Carbon Copied receiver, some hidden receiver or BCC and finally, the body, body of the mail. Along with that we of course, need to use the send buttons to send the mail at the end of the composition.

(Refer Slide Time: 27:22)

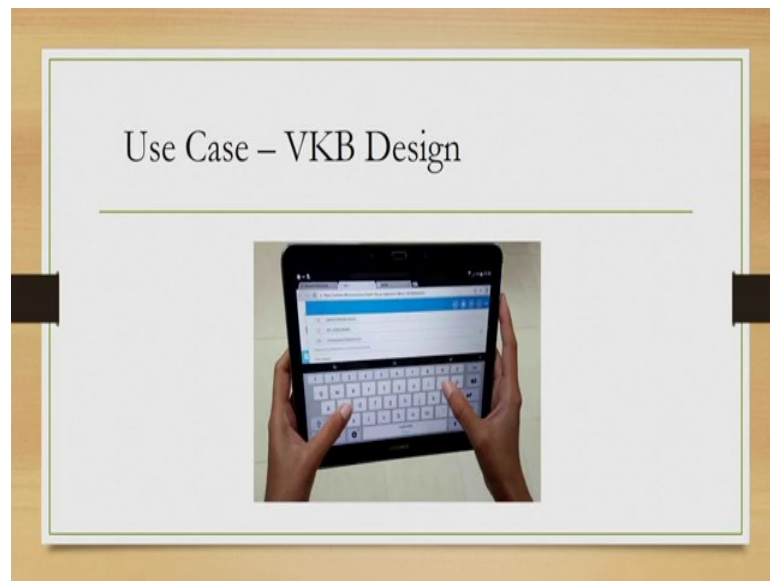


Use Case – VKB Design

- Lots of text entry
- We use virtual keyboards (on phone, tab)

Now, what all these things imply they imply that the text entry related activities, lots of such activities are there and they constitute a very important component of the overall purpose of sending mail. And what we do what we use to perform this entry, we use virtual keyboards or sometimes called soft keywords. Here there are no physical keys so, everything appears on the screen you tap on the keys to compose text, and I think all of you are familiar with composition of email with virtual keyboard.

(Refer Slide Time: 28:11)



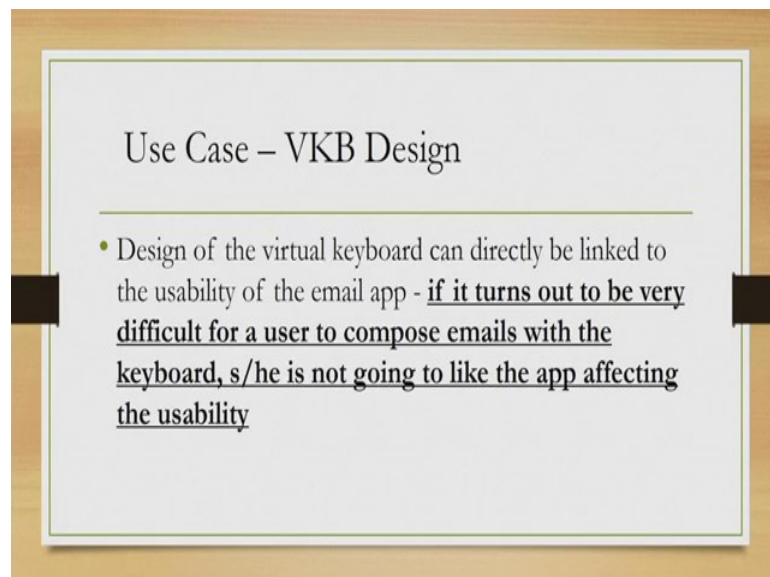
This is one example scenario similar things you probably are familiar with you are doing regularly where we use this virtual qwerty keyboard to input various texts in those fields and after completion we send the email.

(Refer Slide Time: 28:35)



So, clearly it indicates that this virtual keyboard plays a very important role in the overall emailing app or emailing system. The design of the virtual keyboard is likely to determine the usability of the overall app because this particular component of the app is used heavily most of the time.

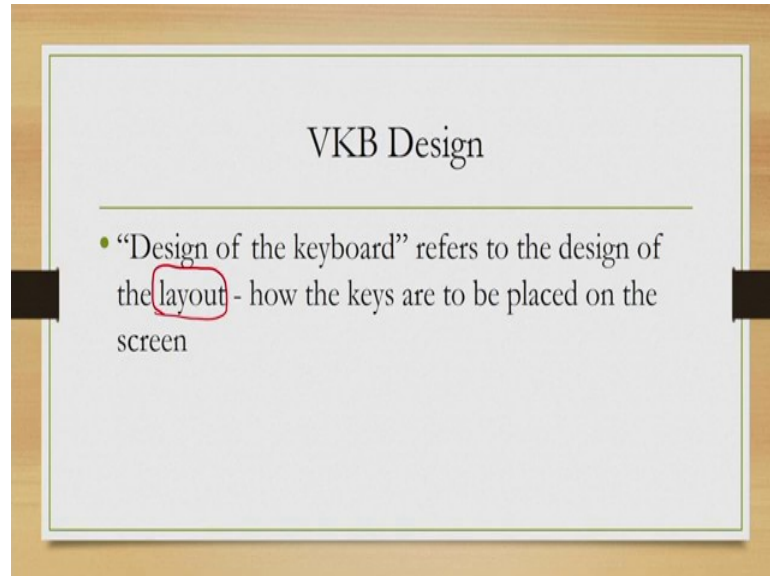
(Refer Slide Time: 29:05)



And if it turns out to be very difficult for a user to compose emails with the keyboard then he or she is not going to like the app which in turn affects usability, acceptability. So, our objective should be to have a design of the keyboard which is usable. So, that the

user finds it very convenient, very easy to compose emails with the keyboard and overall usability of the app increases.

(Refer Slide Time: 29:37)



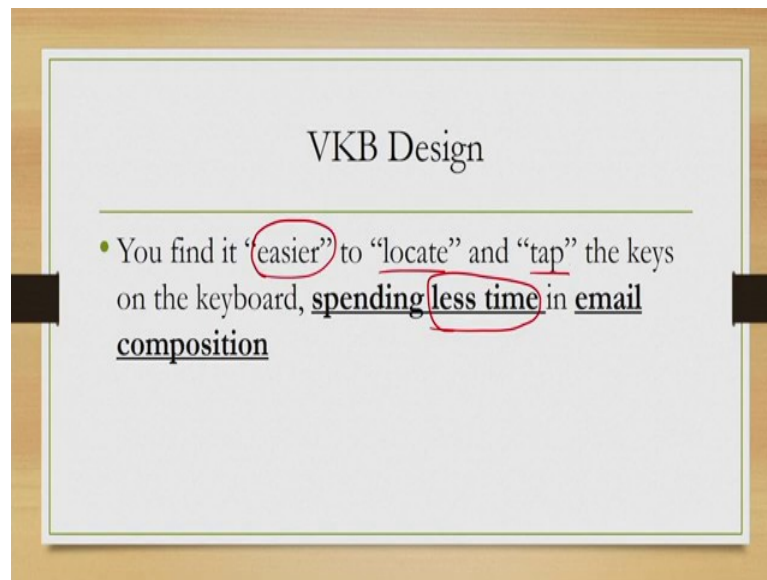
Now, when we talk of design of the keyboard what we actually refer to, it refers to the design of the layout. The geometric arrangement of the keys on the interface around the screen. So, our objective is then to basically come up with the layout that makes text composition very comfortable easy and fast.

(Refer Slide Time: 30:08)



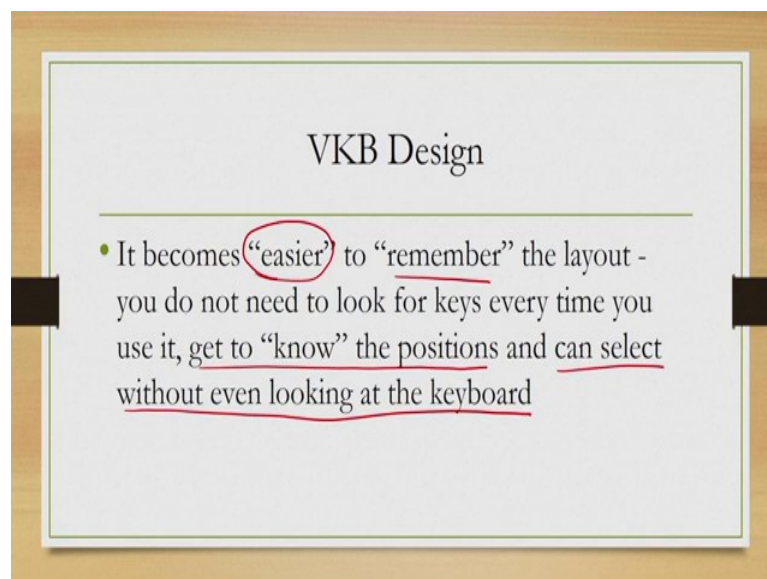
A well designed layout serves many purposes.

(Refer Slide Time: 30:11)



So, to name a few of those purposes first of all you will find it easier and you will spend less time in email composition. So, if you find it easier to locate and tap the keys then definitely were going to take less time to compose your text.

(Refer Slide Time: 30:34)

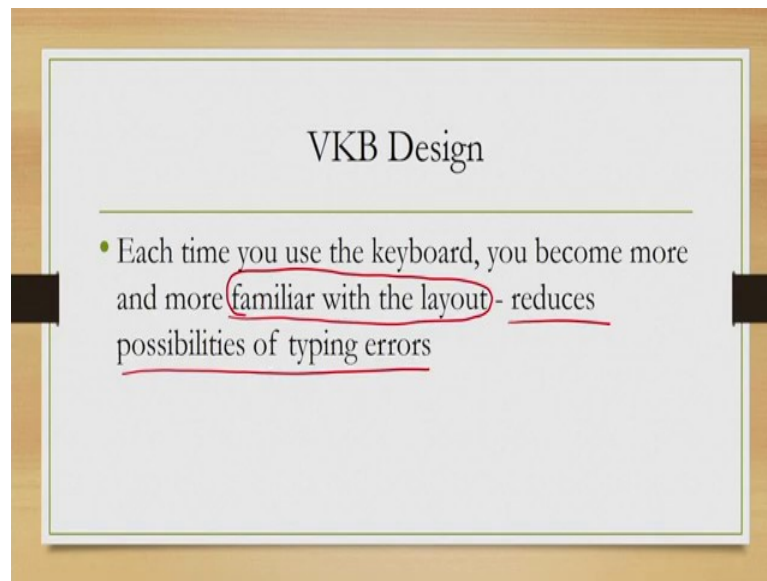


Secondly, it becomes easier to remember the layout. So, it is going to be helpful in again speeding up your next competition task because you do not need to spend time to look at keys for each key selection. You will get to know the positions and memorize them, and can select the keys without even looking at the keyboard, that is typically what we expect

and that is typically once we become expert typist we type without looking at the keyboard.

We inherently remember the key positions and we can basically take our fingers to those positions without even looking at the keyboard. And if you design a proper layout of the virtual keyboard the same purpose can be served and it will speed up the next composition task.

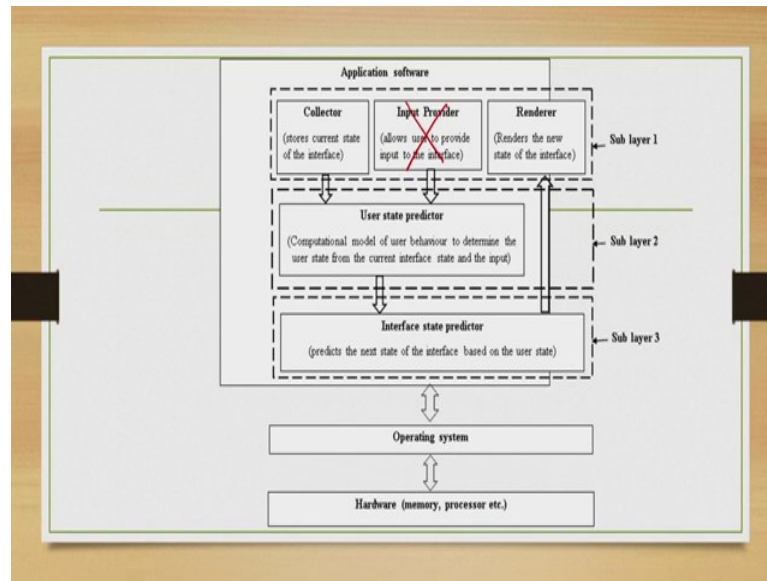
(Refer Slide Time: 31:39)



Thirdly, a good layout reduces possibilities of typing errors as well because it will become familiar if the layout is done in a proper way, systematic way then it is easier for you to become familiar with it and the chances of selecting wrong keys reduces and reduces the overall possibility of typing errors.

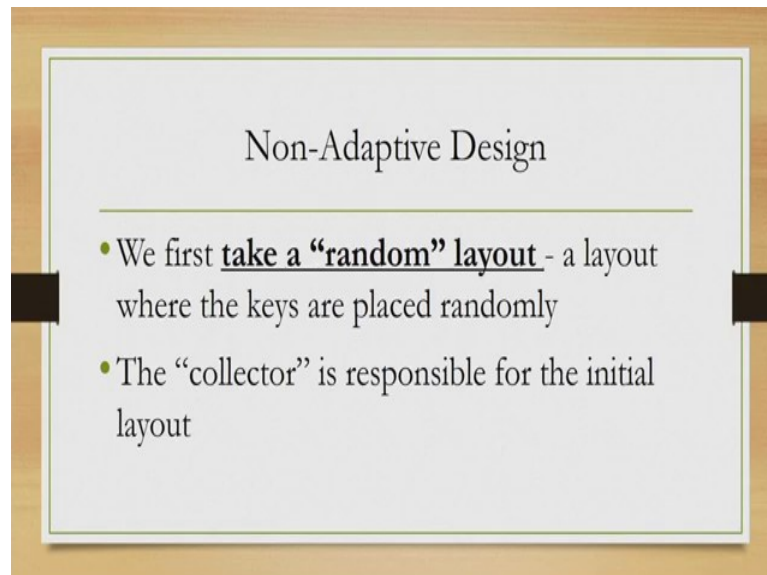
So, it increases familiarity with the layout, if you design the layout in a better way. So, there are other advantages as well, now our objective then is to come up with designs that serves these purposes. So let us first have a look at how we can design this virtual keyboard which is non-adaptive.

(Refer Slide Time: 32:26)



We will try to understand the design in terms of our framework. So, let us have a look at the framework now, were going for a non-adaptive design. So, here we do not require input provider we can discard it, as we have discussed before other components may be useful in understanding the layout, understanding the design of the layout.

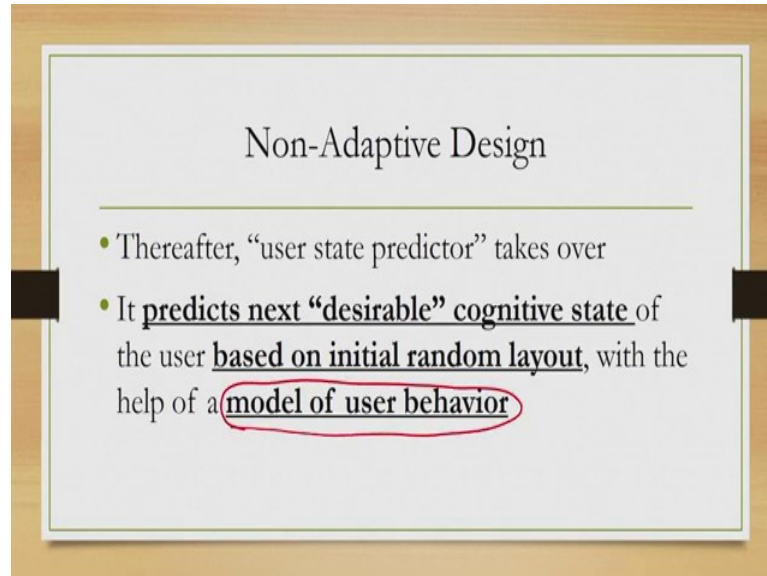
(Refer Slide Time: 32:51)



So, how we can design, let us first have a brief schematic description of the design. So, we start with a random layout. A layout that we generated by randomly placing keys on a on a fixed screen area. Now, this positions the positions of the keys in that random initial

layout is captured by the collector. So, collector is responsible for generating this initial random layout.

(Refer Slide Time: 33:21)



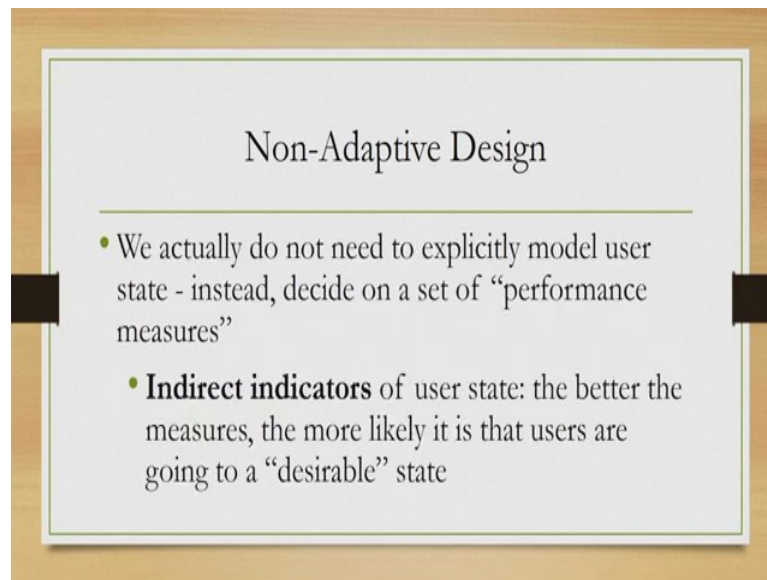
Non-Adaptive Design

- Thereafter, “user state predictor” takes over
- It predicts next “desirable” cognitive state of the user based on initial random layout, with the help of a model of user behavior

Now, once we have that information of the initial layout the collector passes it on to the user state predictor. Now, what the user state predicted does is as we have discussed before the state is defined in terms of some measures performance measures.

And, the user state predictors past computes, those measures for the current state of the interface and based on that it tries to predict a state which actually improves those measures which is the desirable next state and it passes it on to the sub layer three that is interface state predictor. Now, in order to predict the next desirable state, what is required is a model of user behavior; now that is again something new which we are introducing at this stage.

(Refer Slide Time: 34:23)

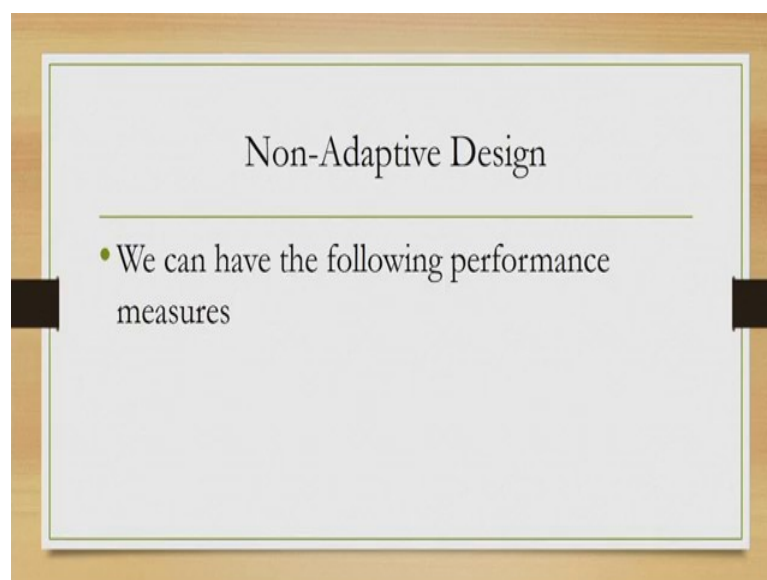


Non-Adaptive Design

- We actually do not need to explicitly model user state - instead, decide on a set of “performance measures”
 - **Indirect indicators** of user state: the better the measures, the more likely it is that users are going to a “desirable” state

So, what we mean by the model is essentially that when a keyboard is presented to the user how the user is likely to type and based on that we can compute the performance, and based on that we can compute the current state, and based on that we can compute the next state. So, this state as we have said is comprising of a set of performance measures which are indirect indicators of the user state.

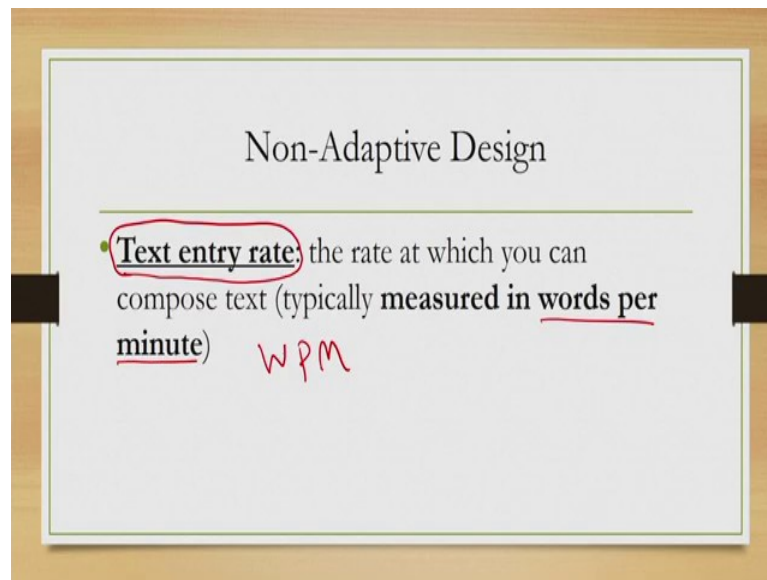
(Refer Slide Time: 35:01)



Non-Adaptive Design

- We can have the following performance measures

(Refer Slide Time: 35:02)

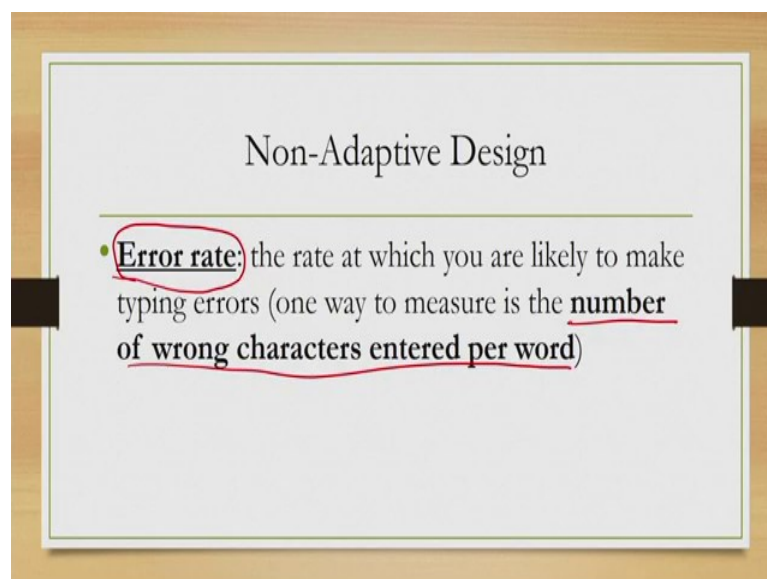


Non-Adaptive Design

- Text entry rate: the rate at which you can compose text (typically measured in words per minute) WPM

Now, for a virtual keyboard we can have various performance measures, the common ones that are typically used. The first one is text entry rate that is the rate at which you can compose text. So, this is the first and most obvious and most common performance measure used for measuring the performance of a typist with a keyboard, and the typical unit of measurement is words per minute or wpm. Sometimes, we have words per second also, characters per seconds also, but word per minute is the most common measure..

(Refer Slide Time: 35:50)

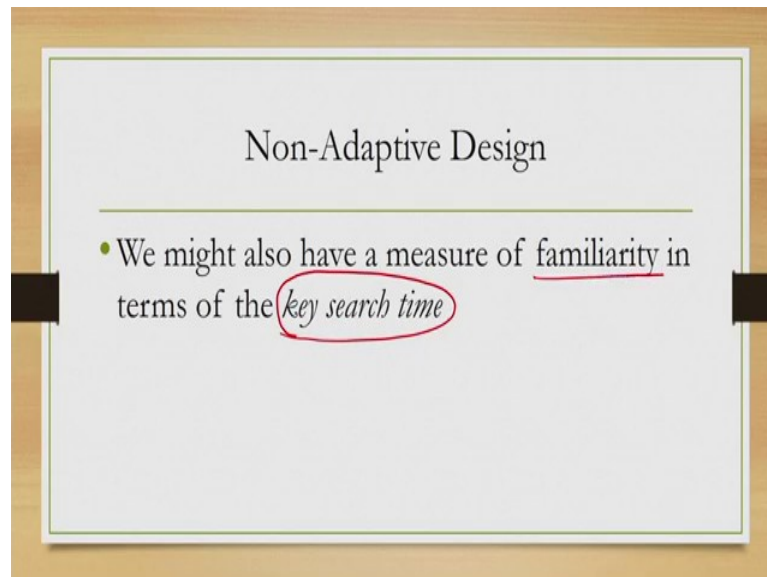


Non-Adaptive Design

- Error rate: the rate at which you are likely to make typing errors (one way to measure is the number of wrong characters entered per word)

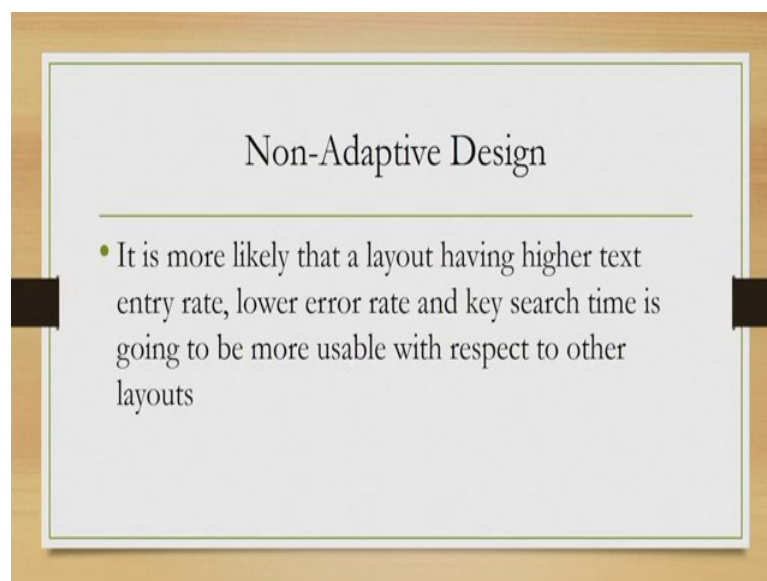
Second measure can be error rate. So, how much errors the users are making? So, it is measured in terms of some rate, one such measure can be number of wrong characters entered per word. So, in order to compose a word how many wrong characters you have entered, there can be many other similar ways to define this error rate.

(Refer Slide Time: 36:19)



There can be third measure that is the key search time, how much time it takes for a user to look at a key on the screen, it is also an indirect measure of familiarity.

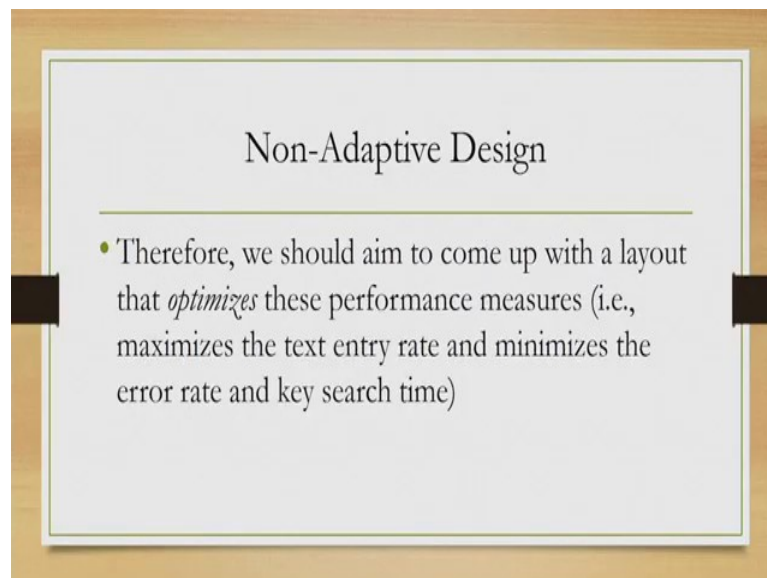
(Refer Slide Time: 36:31)



Now, when we talked of coming up with the next desirable state which improves performance what will meant; essentially, what we meant something like this in the current state we have a random layout based on this layout we have a model of users typing behavior. We apply that model to come up with values compute the values of the measures such as entry rate, error rate, search time.

We get a set of values which defines the current state of the user and we try to generate a layout which improves these values which improves a text entry rate, reduces error rate, reduces search time. If we get such a state that is clearly a desirable state that improves the typing performance and we want to we want the user to go to that state or have that state while typing. So, that is our computed next step.

(Refer Slide Time: 37:37)



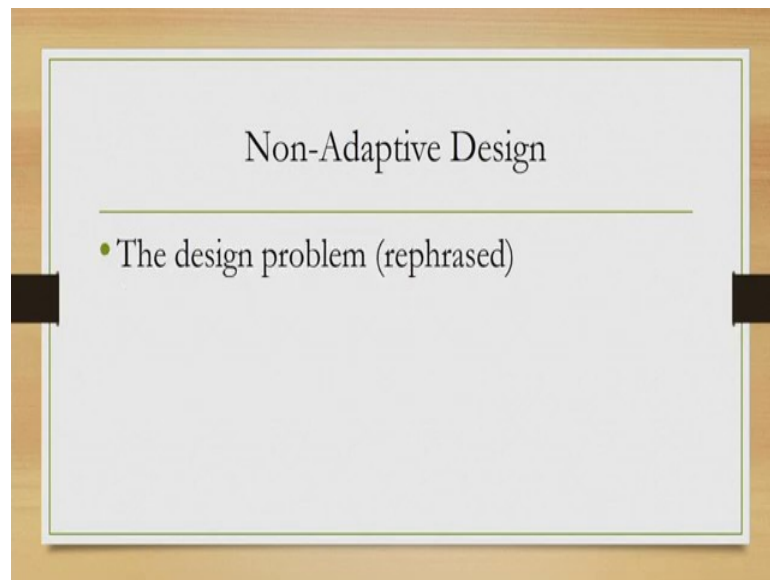
The slide features a light gray background with a thin green border. The title 'Non-Adaptive Design' is centered at the top in a black serif font. Below the title is a horizontal green line. A single bullet point is positioned below the line, containing text in a black serif font. The slide is set against a light brown background with two black rectangular accents on the left and right sides.

Non-Adaptive Design

- Therefore, we should aim to come up with a layout that *optimizes* these performance measures (i.e., maximizes the text entry rate and minimizes the error rate and key search time)

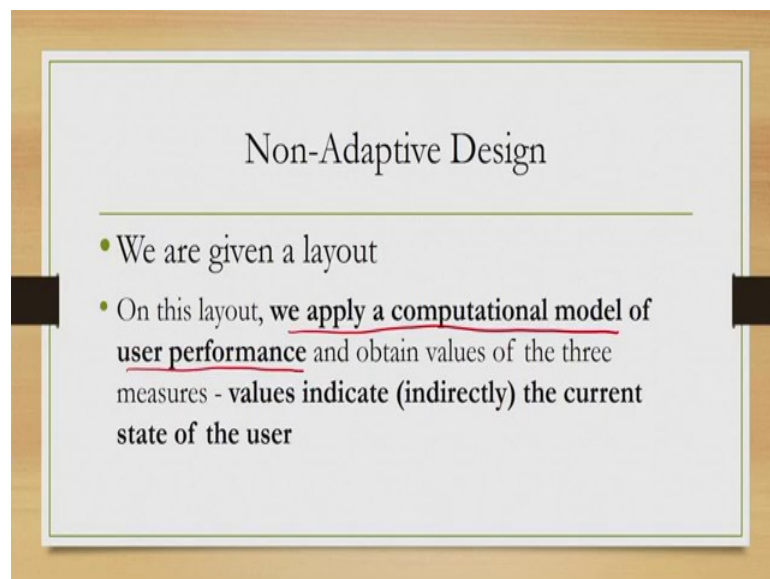
So, our overall objective then is to come up with a layout starting from an initial random layout that optimizes these measures, in other words increases text entry rate and decreases or reduces error rate and search time, and that optimize layout is likely to be more usable compared to any random layout.

(Refer Slide Time: 38:07)



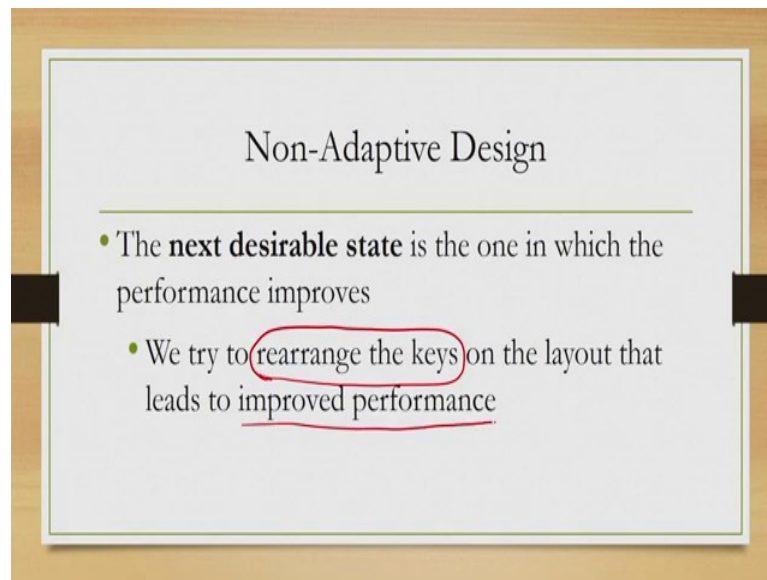
So, then we can rephrase the design problem.

(Refer Slide Time: 38:10)



As we are given a layout and on this layout we apply a computational model of user performance, and obtain values of the measures these values indicate indirectly the current state of the user.

(Refer Slide Time: 38:30)

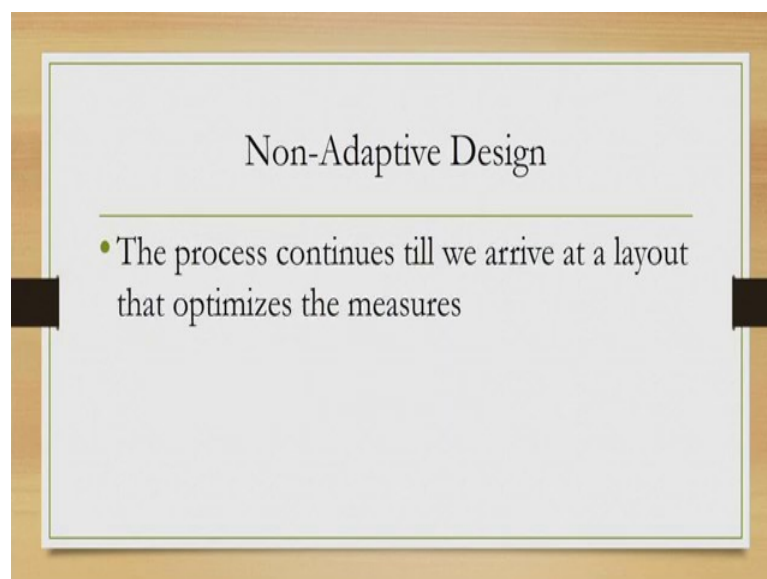


Non-Adaptive Design

- The **next desirable state** is the one in which the performance improves
- We try to rearrange the keys on the layout that leads to improved performance

And the next desirable state is the one in which we improve or in which we try to improve the measures which indicates that the performance over all text entry performance improves that we can do by rearranging the keys on the layout. Clearly, the initial layout give some measures in order to improve it we need to change the layout. So, we have to rearrange the keys on the layout which is likely to lead to improved performance.

(Refer Slide Time: 39:11)



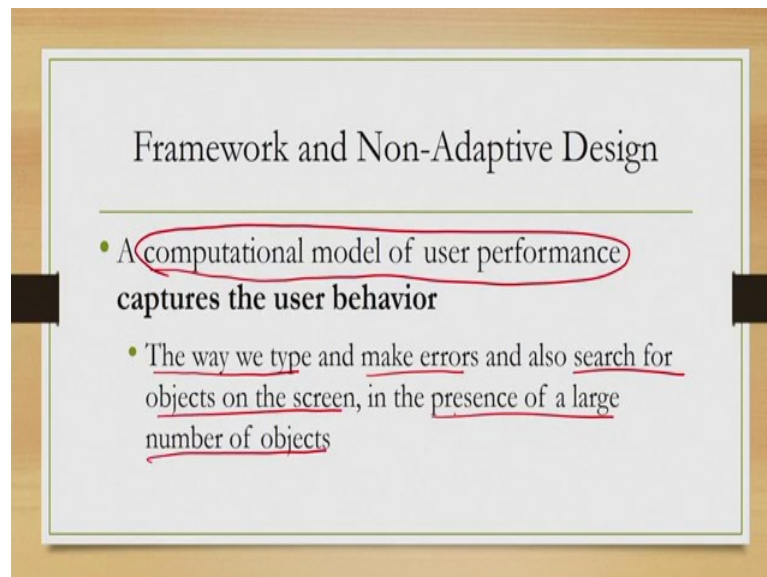
Non-Adaptive Design

- The process continues till we arrive at a layout that optimizes the measures

So, this is an iterative process. So, we start with the random layout, compute using the computational model of typing behavior, compute the measures, get a set of values then rearrange the layout again compute, compare it with the previous measures.

See whether there is some improvement if yes then we try to see how significant that improvement is sufficient free significant and there is no scope for further improvement we stop; otherwise, we again change the layout, again compute the performance measures and compare it with the previous measures and this process continues. So, it is an iterative cycle and the cycle continues till we reach an optimum layout.

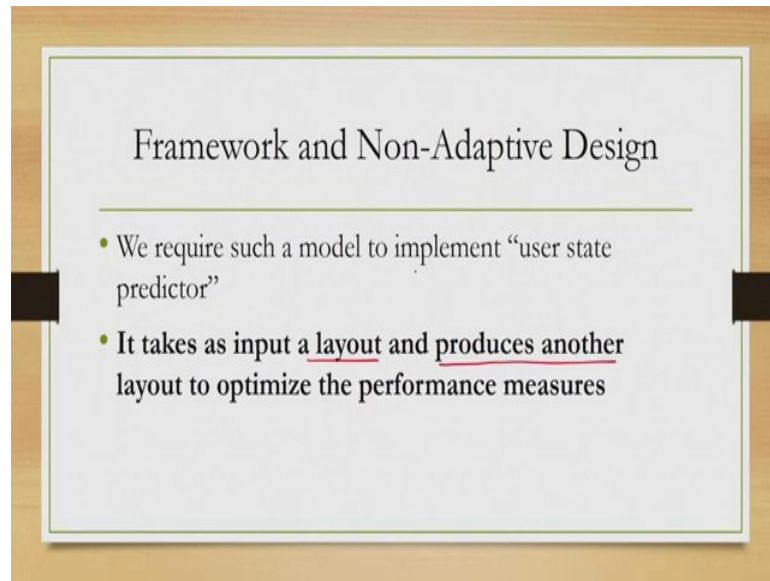
(Refer Slide Time: 39:55)



So, the key concern here is computational model of user performance, unless we have it unless we have a computational model of performance we cannot capture the behavior, we cannot compute the measures. Now, what this model indicates or represents, it represents the way we type, make errors, search for objects on the screen, in the presence of large number of objects..

Clearly, this refers to a keyboard having large number of keys and how we type on a keyboard, how we make errors, how we search for objects, essentially our typing behavior. And unless we have such a model we cannot implement the user state predictor.

(Refer Slide Time: 40:51)



Framework and Non-Adaptive Design

- We require such a model to implement “user state predictor”
- It takes as input a layout and produces another layout to optimize the performance measures

So, this model takes as input. So, earlier we said user state predictor it is essentially and implementation of a model that takes as input a layout and the produces another layout to optimize the performance measures. Our objective is to have a model to implement the state predictor, it takes as input a layout and produces another layout to optimize the performance measures. From where to get that model? That is one important question. Fortunately, we have such models and in later lectures we will discuss those models.

(Refer Slide Time: 41:39)

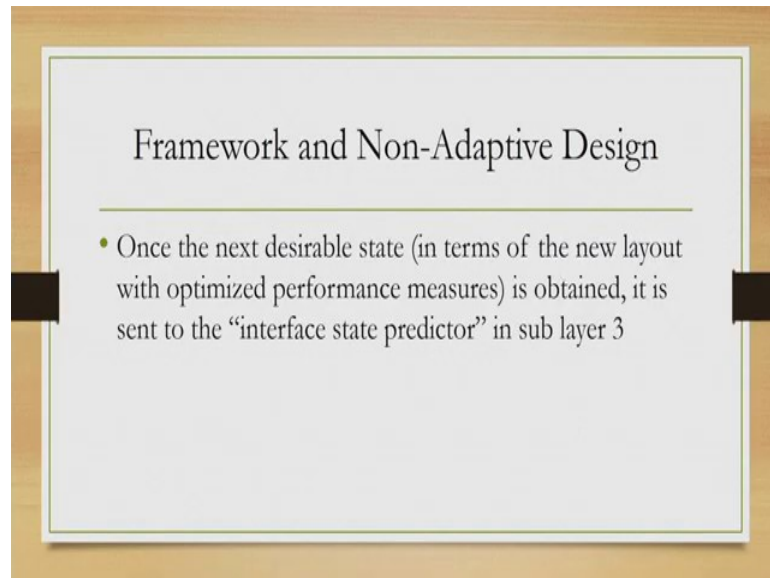


Framework and Non-Adaptive Design

- **Good news - we do have such a model of typing behavior (to discuss later)**

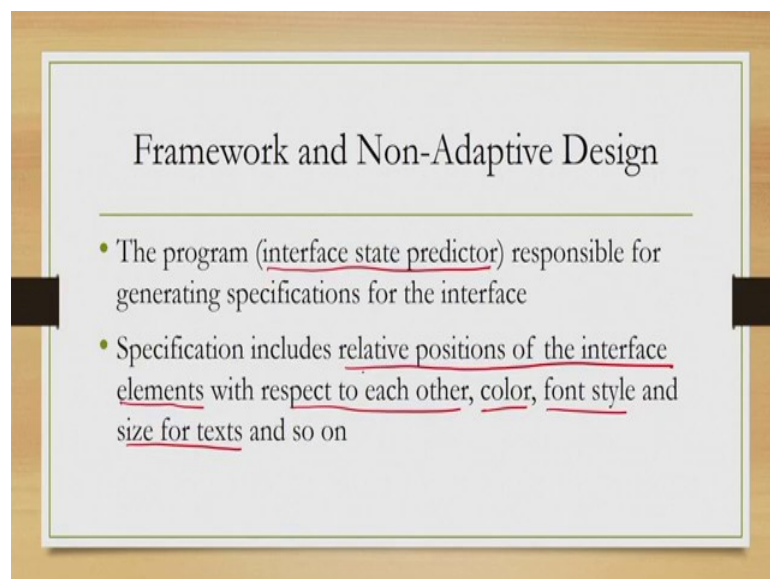
For the time being we ignore the details of the models and we will assume that we do have such model available to us.

(Refer Slide Time: 41:53)



Now, once the model manages to come up with the next desirable state that information is passed on to the interface state predictor which produces specification of the interface that conforms to the next desirable state.

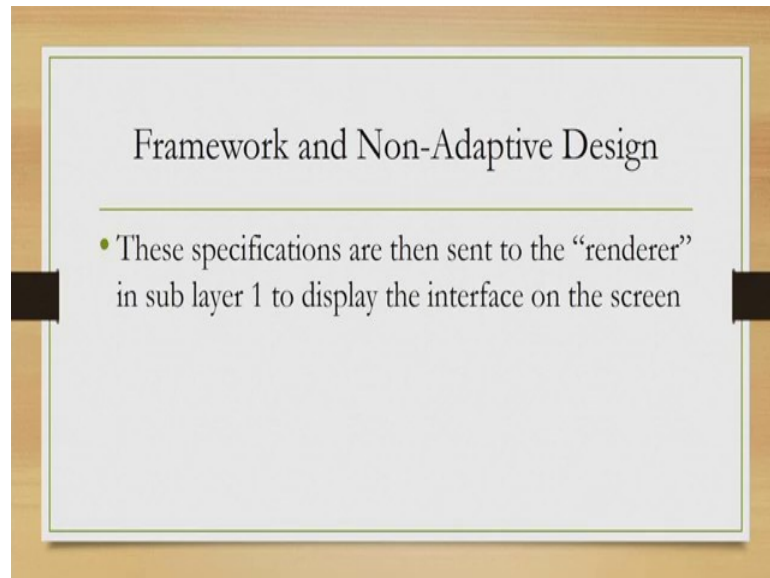
(Refer Slide Time: 42:14)



So, in other words its job is to produce a layout that conforms to the desirable state requirement. And, this layout specification includes or may include relative position of

the interface elements that is the keys and other buttons with respect to each other, the color, the font style, text size so on. So, the job of the interface state predictor is essentially to come up with this specifications.

(Refer Slide Time: 42:53)



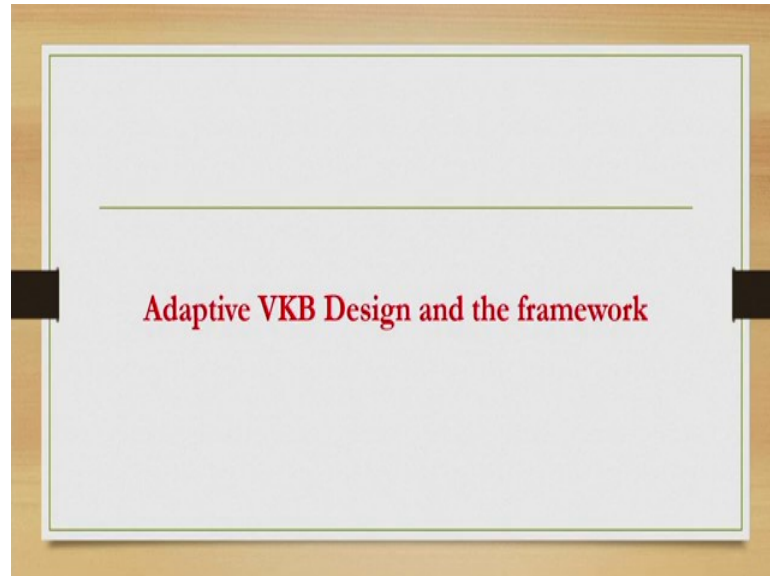
And once those specifications are available it is sent to the render which based on the specification renders the keyboard on the screen, then we have non adaptive design. Now, let us briefly talk about an adaptive design.

(Refer Slide Time: 43:18)



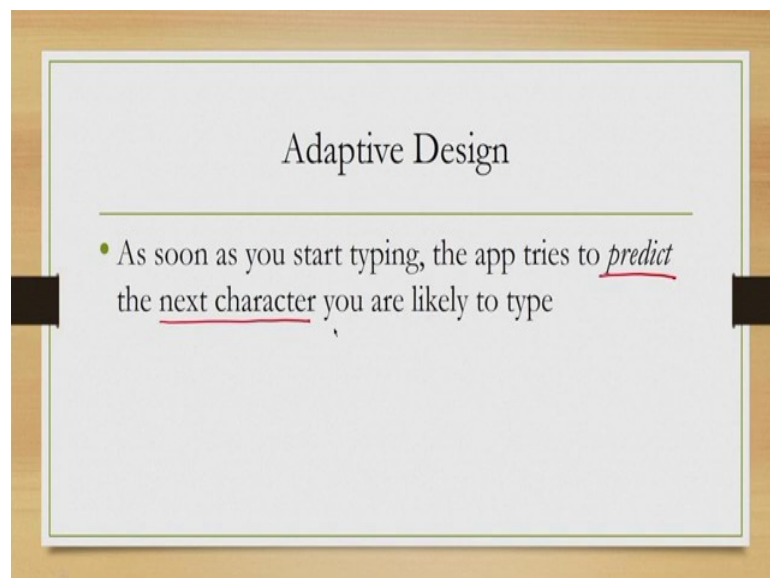
How we can think of adaptive keyboard with the help of the framework the computational framework?

(Refer Slide Time: 43:23)



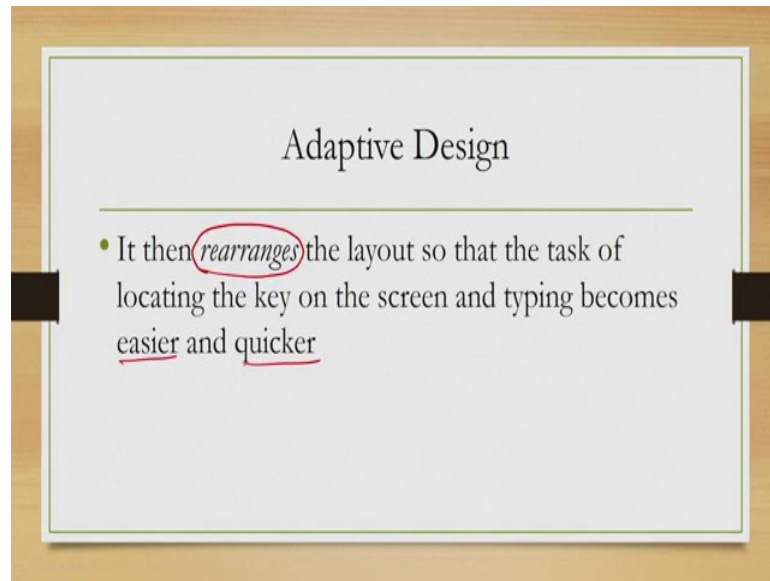
Now, when we are talking of adaptive design. So, what we mean is that the app comes with a default layout which may or may not be optimized.

(Refer Slide Time: 43:36)



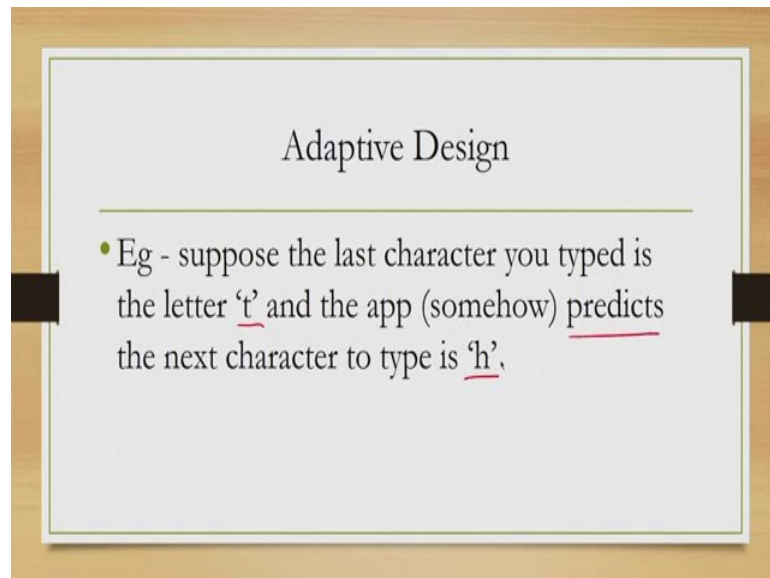
As soon as you start typing the app tries to predict the next character you are likely to type. So, prediction of next character is important; so, as soon as you start typing that tries to predict what are the next characters you are likely to type.

(Refer Slide Time: 43:57)



And it then rearranges the layout, changes the layout so, that the task of locating the key on the screen and typing becomes easier and quicker. So, essentially the keys are placed in such a way so, that you will find them just beside the currently typed key and it does not take you time to look at it

(Refer Slide Time: 44:26)



For example, suppose the last character you have typed is the letter 't' and the app somehow manages to predict that the next character you are likely to type is 'h'.

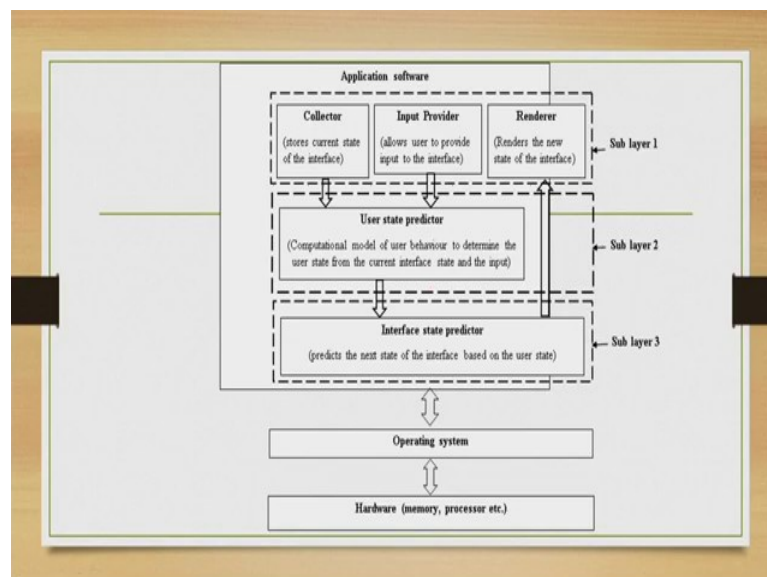
(Refer Slide Time: 44:42)

Adaptive Design

- The app rearranges the layout and puts 'h' just beside the last letter you typed (that is 't'), since that is where your finger currently is
- You save on both the key search time and the finger movement time, increasing typing speed

So, it rearranges the layout and puts h just beside the last letter you have typed that is t, since that is where your finger currently is and then what is the advantage then you say both the key search time at the finger movement time. So, both are reduced and which in turn increases the typing speed you do not need to spend more time on locating the key and moving your finger to that key.

(Refer Slide Time: 45:15)

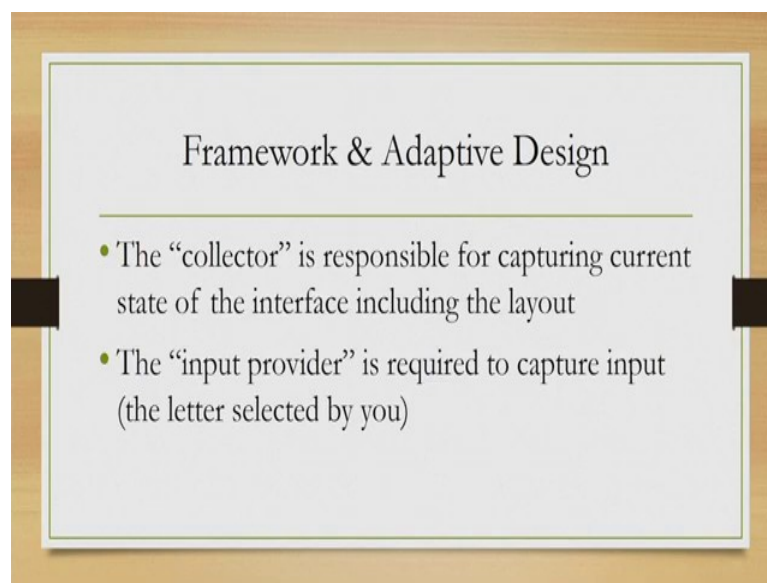


So, here let us recollect the framework as you can see here all the components are required we require input provider which will capture the keys typed, require collector

which will capture the current layout, these will be sent to the user state predictor which will predict based on the computational model.

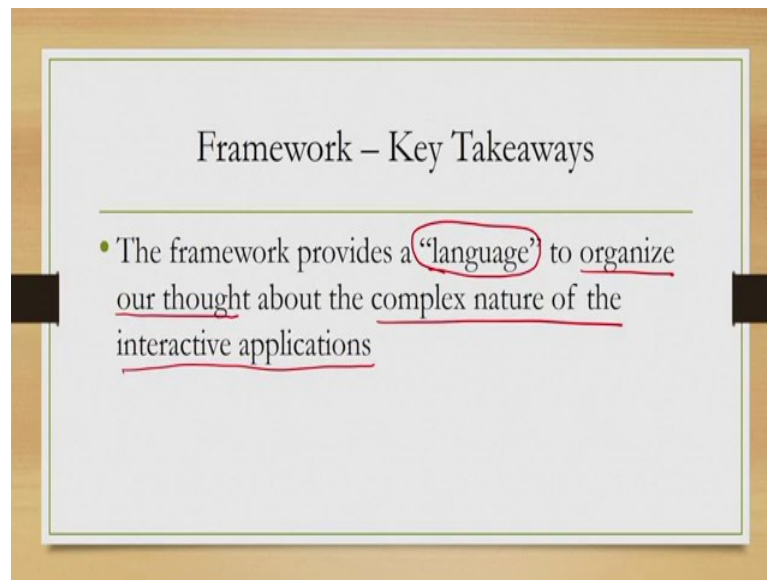
The next desirable state and based on that information the interface state predictor will come up with a layout which conforms to that state and that layout specification will be sent to the renderer where the render will render it and display it, and this is not one time. So, this will keep on happening when you are actually using the key. So, it will keep on happening during your usage of the keyboard or during the execution time.

(Refer Slide Time: 46:11)



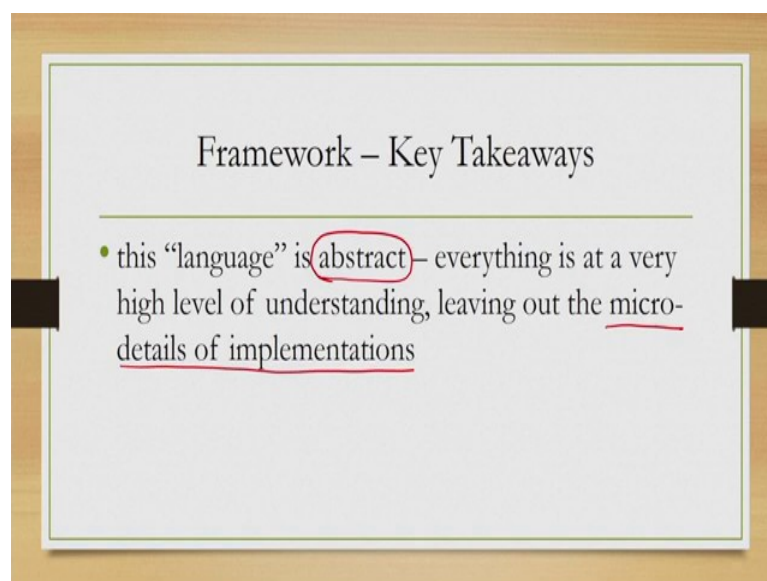
So, all the components we need to implement to be able to come up with a adaptive keyboard layout.

(Refer Slide Time: 46:22)



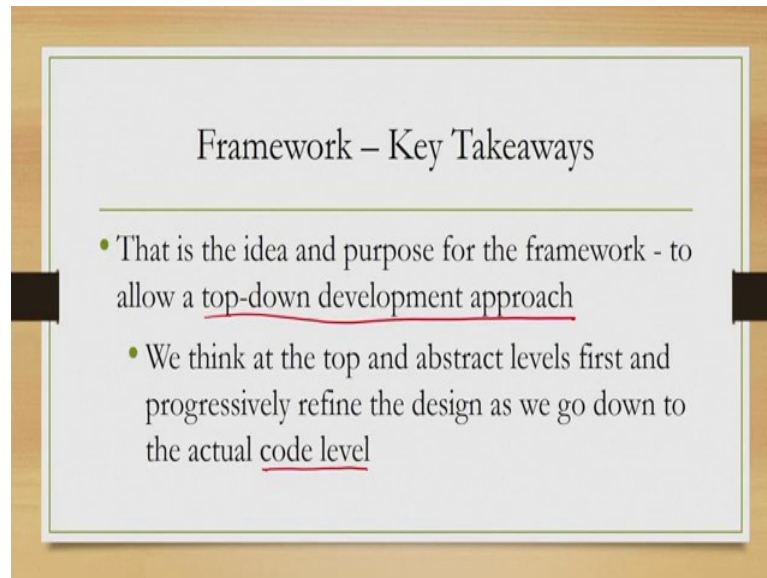
So, these two examples are meant to give you some idea of how to relate the framework to the final design. So, what are the key takeaways from this discussion. First of all the framework provides a language to organize our thought. So, it is a high level concept, it helps us to organize our thought by providing a language, thought about complex nature of the interactive application. Any interactive application is very complex and we needed some way to understand it, some way to represent it and this framework gives such away, framework provides a language to understand that.

(Refer Slide Time: 47:13)



The language is abstract at a very high level of course, as you can see the micro details of implementations are not there which is left to the imagination of the designer. So, the language is very flexible in the sense that the micro details can be worked out by the designer based on his or her expertise knowledge intuition, but that working out will proceed along a more systematic path we service if we do not have this framework.

(Refer Slide Time: 47:55)

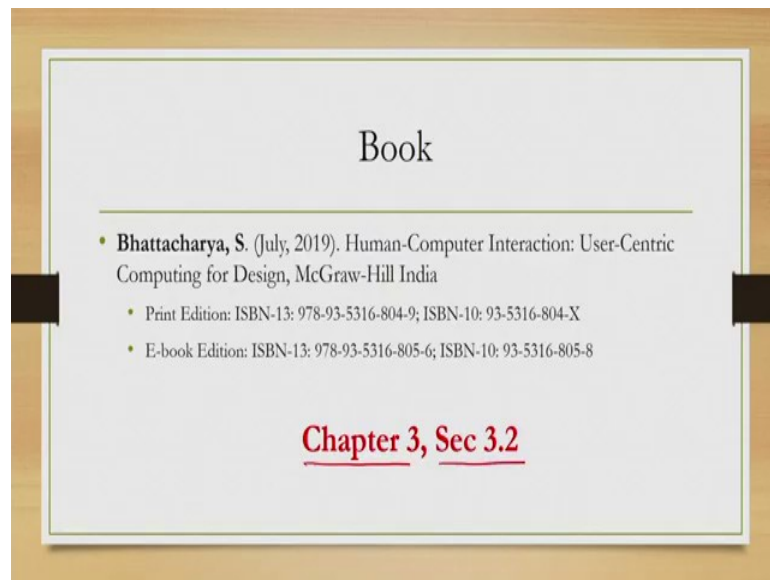


So, that is the overall idea of the framework that it is meant to allow a top down development approach. So, essentially you think of the system at a very top or abstract level and then keep on working out the subsystems starting from that abstract level view and then try to implement at the code level. So, this top down development approach is the key purpose and objective of the framework.

Whatever we have discussed today, let me just recollect once. So, we talked about a framework which is mean to give you, some idea some structured way of thinking about the complex interactive systems. There are three sub layers in the framework, most important layer is the user state predictor which comprises of a model computational user model. And, we tried to understand this framework in terms of two examples they belong to two broad categories of interactive systems adaptive and non adaptive.

So, we tried to illustrate the idea by explaining one non-adaptive virtual keyboard design and one adaptive virtual keyboard design by relating the design to the framework with the overall objective of explaining the idea of the framework.

(Refer Slide Time: 49:33)



So, whatever discussion we had today are based on chapter 3 section 3.2 of this book Human-Computer Interaction: User-Centric Computing for Design.

Thank you and goodbye.