User-Centric Computing for Human-Computer Interaction

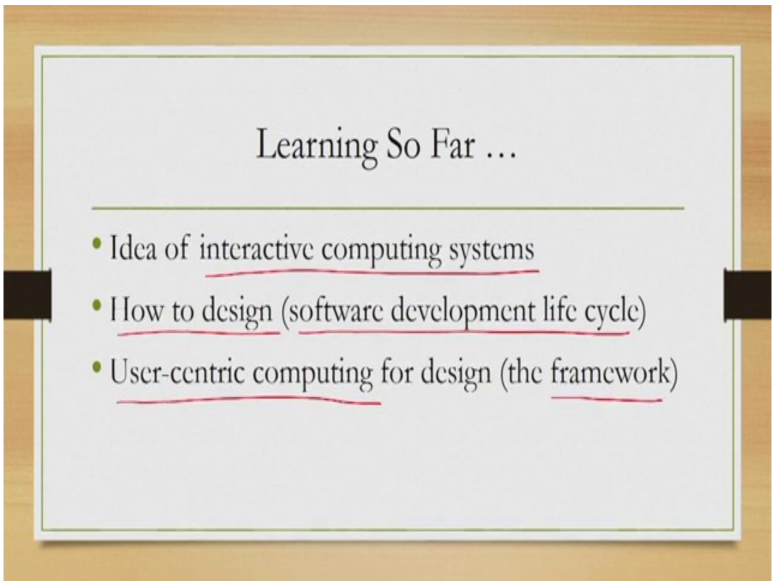
Prof. Samit Bhattacharya
Department of Computer Science and Engineering
Indian Institute of Technology, Guwahati

Lecture - 25
Case studies on the use of models

Hello and welcome to lecture number 25 in the course User - Centric Computing for Human - Computer Interaction. Till the last lecture we have covered many topics related to user models more specifically computational user models. So, before we proceed to the next topic it is time to take a break and try to understand in a more detailed way how to use these models in the design of overall interactive system.

So, in this lecture today we are going to see a few Case studies to understand the use of the models in the design of interactive systems. Note that earlier we also mentioned this point; however, we mentioned it at a very shallow level today we are going to discuss it in more details with illustrative case studies. Before that as is the convention we will quickly recap what we have learnt so far and then we will proceed to the main discussion for today's lecture.

(Refer Slide Time: 02:00)

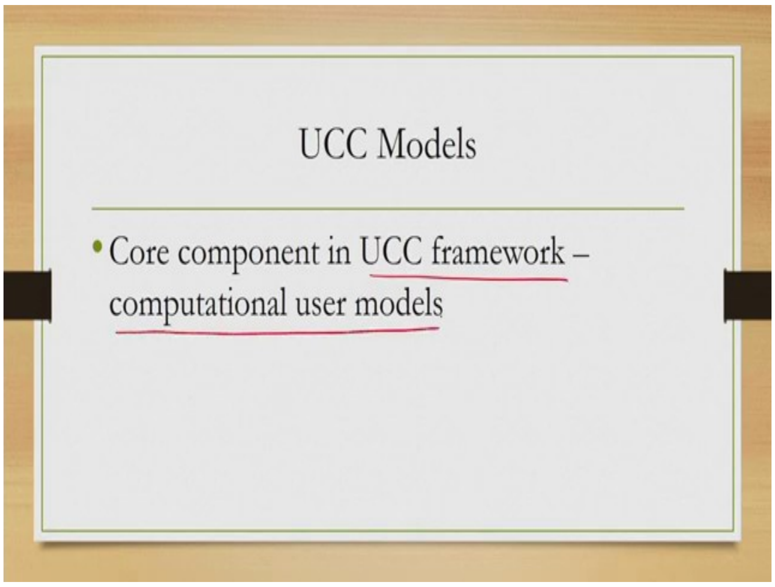


Learning So Far ...

- Idea of interactive computing systems
- How to design (software development life cycle)
- User-centric computing for design (the framework)

So, if we recollect so far in the previous 24 lectures we have learned about the idea of interactive computing systems, a broad idea. This was followed by how to design an interactive computing system, more specifically we focused on the use of software development life cycle to build interactive computing systems. This was followed by discussion on the idea of user centric computing for design, where we mentioned one framework.

(Refer Slide Time: 02:39)

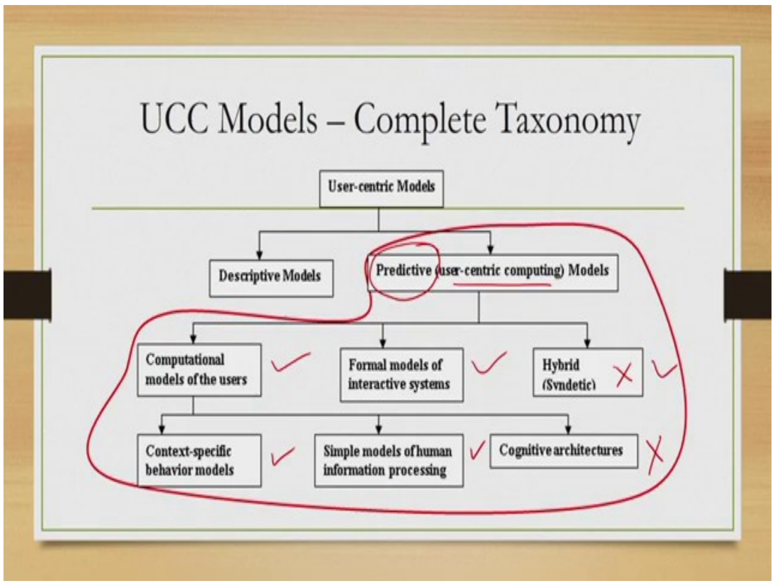


UCC Models

- Core component in UCC framework – computational user models

Along with that we also mentioned that the core component of the framework is the computational user models.

(Refer Slide Time: 02:48)



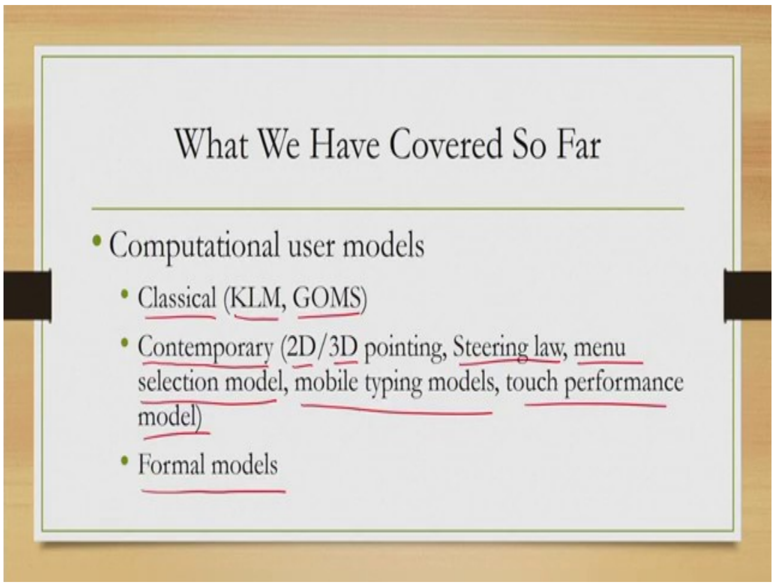
UCC Models – Complete Taxonomy

User-centric Models
Descriptive Models
Predictive (user-centric computing) Models
Computational models of the users
Formal models of interactive systems
Hybrid (Syndetic)
Context-specific behavior models
Simple models of human information processing
Cognitive architectures

And we had some introductory discussion on different types of such models, where we identified a class of models that is our main focus those are called predictive models or user centric computing models. Under the predictive models there are several subcategory of models we have computational models of the users, formal models and hybrid models, hybrid models we did not discuss. Under computational user models we have context specific behavior models, simple models of human information processing and cognitive architectures.

Like hybrid models we did not discuss cognitive architectures both these categories are too complex and out of scope in our particular focus area. We have already discussed several models that belonged to either computational user or formal models.

(Refer Slide Time: 03:47)



What We Have Covered So Far

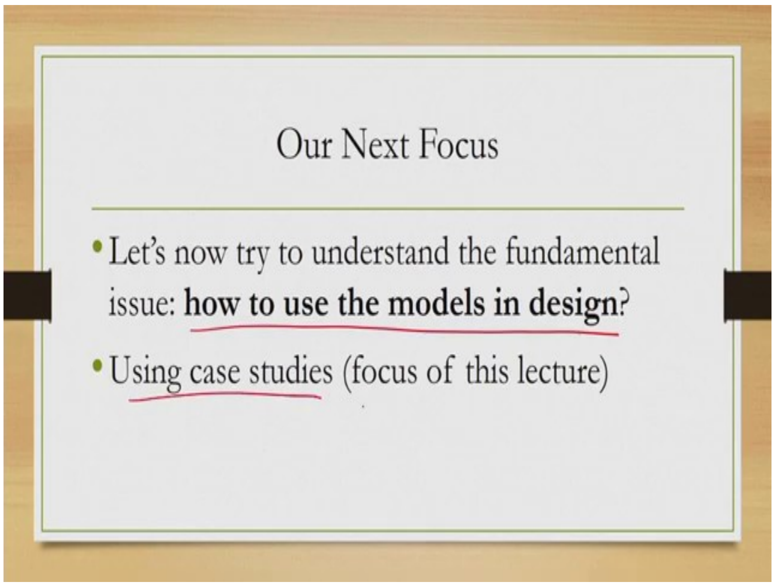
- Computational user models
  - Classical (KLM, GOMS)
  - Contemporary (2D/3D pointing, Steering law, menu selection model, mobile typing models, touch performance model)
  - Formal models

So, under this computational user models what we have discussed, we have discussed classical models as well as contemporary models. Under the classical models we have discussed the keystroke level model and the CMN GOMS model, both belonging to the overall GOMS family of models, in addition we have discussed the Fitts law and Hick Hymans law.

In contemporary models that is the category of models that have been developed to design systems that we see around us at present, we have discussed 2 D and 3 D pointing models also called the bivariate and tri variate pointing models. The Steering law, which is used for modeling constraint navigation and behavior, the menu selection model modeling the behavior of selecting from a hierarchical menu, mobile typing models and touch performance model or the Fitts law. In the mobile typing model who discussed 2 models namely the Fitts diagraph model and the thumb typing model.

Along with the computational user models we also discussed formal models, essentially we discussed about the use of formal languages formal specification languages to represent dialogues. In particular we have discussed one language that is the state chart and we briefly introduced the idea of using the language to build models of interactive systems using state chart and also how to use the specification to verify properties.

(Refer Slide Time: 05:26)

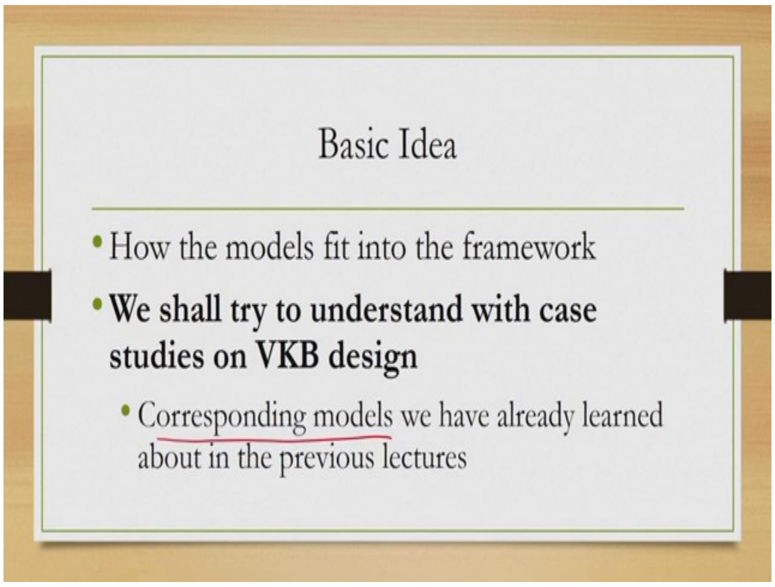


Our Next Focus

- Let's now try to understand the fundamental issue: **how to use the models in design?**
- Using case studies (focus of this lecture)

So, we have got some fairly good idea of what are these models and for what purpose they are used. Now, let us try to understand one fundamental concern here that is, how to use the models in the design. So, the models that we have discussed were not part of any system when we discuss those models we discussed them in isolation. Now, it will be helpful for you to understand the use of these models in the overall software development process. In order to understand how the user centric computing can help in developing interactive systems software and we will do that in this lecture using case studies for illustration. So, our case study will concern design of virtual keyboards.

(Refer Slide Time: 06:17)

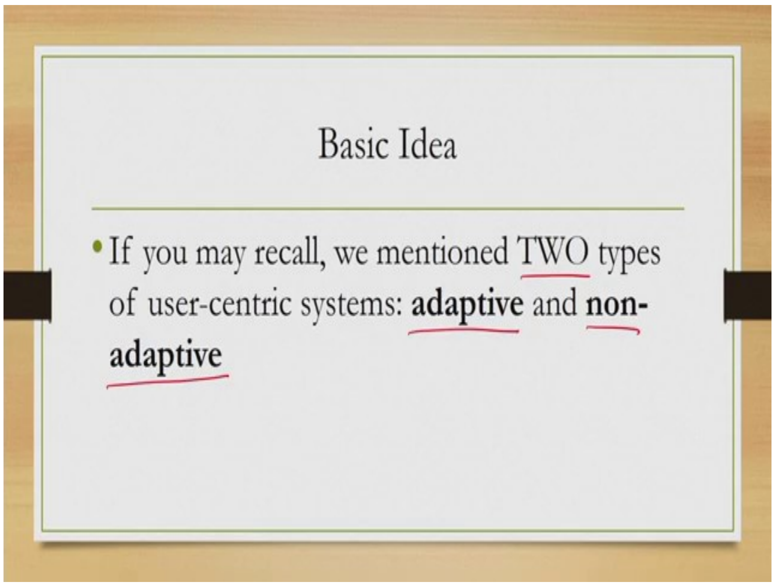


Basic Idea

- How the models fit into the framework
- **We shall try to understand with case studies on VKB design**
  - Corresponding models we have already learned about in the previous lectures

Why we have chosen this case study? Because, we have already discussed the corresponding models in the previous lectures, if you recollect two models we have discussed the Fitts digraph model and the thumb typing model. Now these models will be helpful in understanding how we can use models in order to build an interactive system in this case a virtual keyboard and we will discuss this in terms of few example keyboards.

(Refer Slide Time: 06:49)

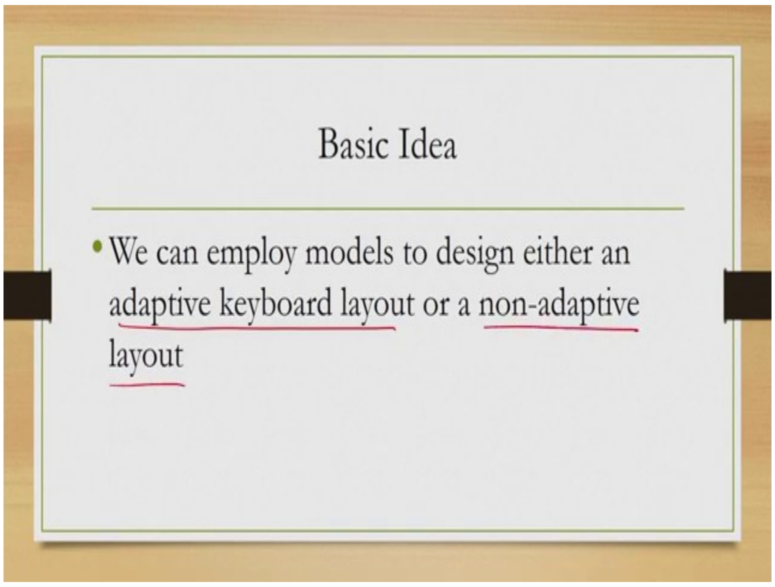


Basic Idea

- If you may recall, we mentioned TWO types of user-centric systems: **adaptive** and **non-adaptive**

If you may recall there are two types of user centric systems that we have discussed earlier; one is the adaptive system and other one is the non adaptive system. In adaptive system we mentioned that when we interact something changes either on the interface or the way we interact whereas, in the case of non adaptive systems it is designed once and the design remains the same throughout the usage of the system.

(Refer Slide Time: 07:17)

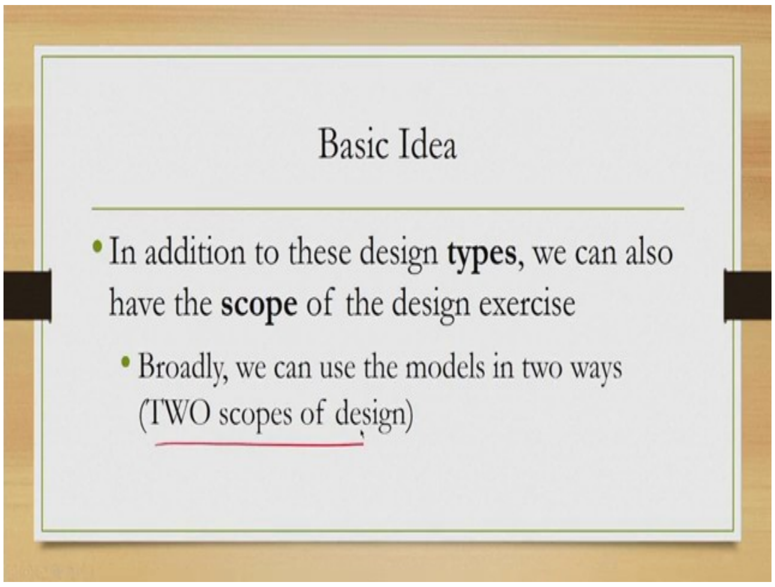


Basic Idea

- We can employ models to design either an adaptive keyboard layout or a non-adaptive layout

Now, the models that we have discussed earlier can be used to design either an adaptive keyboard layout or a non adaptive layout. So, for both purposes we can use the model.

(Refer Slide Time: 07:30)

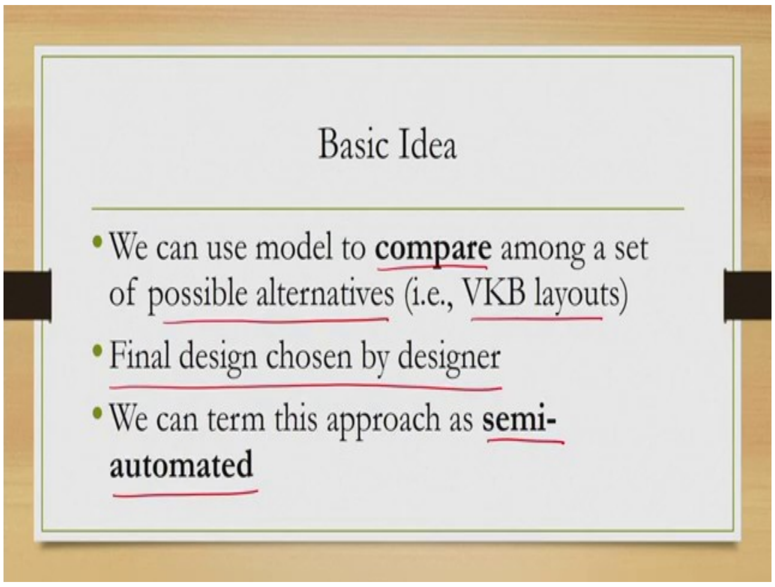


Basic Idea

- In addition to these design **types**, we can also have the **scope** of the design exercise
  - Broadly, we can use the models in two ways (TWO scopes of design)

Along with the types of the designs these two types adaptive and non adaptive designs we can also define scopes of the design accessories broadly two scopes can be defined.

(Refer Slide Time: 07:48)

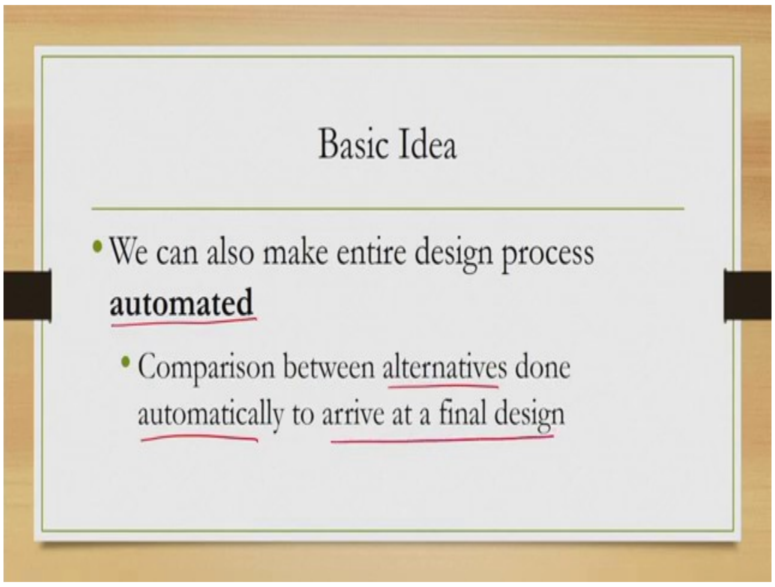


Basic Idea

- We can use model to **compare** among a set of possible alternatives (i.e., VKB layouts)
- Final design chosen by designer
- We can term this approach as **semi-automated**

What are these two scopes? One is semi automated design approach what this means, it means that we can use the models to compare among a set of possible alternatives. So, in our case in this particular case study we are concerned about virtual keyboard layouts. So, there is a set of alternative layouts and we can compare using the models. The result of the comparison is analyzed or inspected by the designer who then chooses the final design. So, final design is chosen by the designer, this is the semi automated approach.

(Refer Slide Time: 08:28)



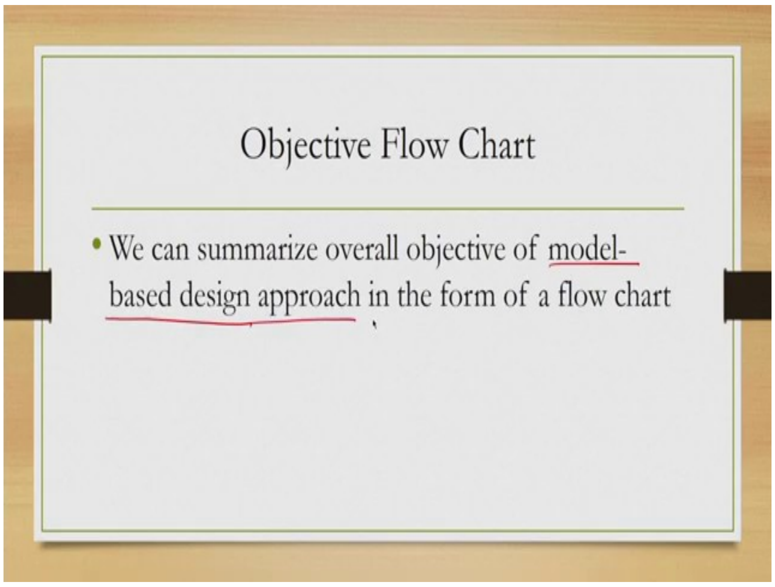
Basic Idea

- We can also make entire design process **automated**
  - Comparison between alternatives done automatically to arrive at a final design

We can also make the entire design process automated. That means the comparison between alternatives is done automatically by the system itself to arrive at a final design so, here the involvement of the designer is minimal. So, these are the scopes, one is semi automated where we are actually relying on the designers judgment. So, the models are used to compare and produce some comparison results and based on the result the designer chooses the best alternative in a given set of alternatives.

This process is automated fully or mostly in the automated design approach where these comparison and the final design decision is taken by the system itself with minimal intervention by the designer.

(Refer Slide Time: 09:19)

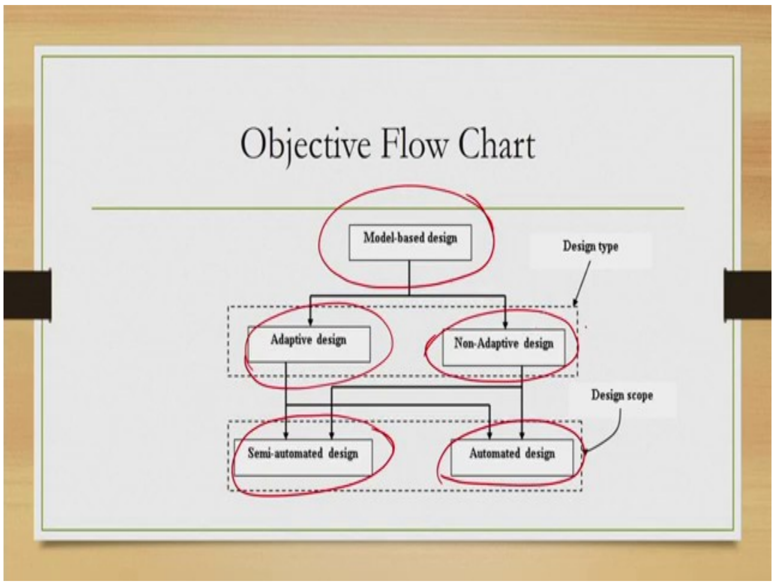


Objective Flow Chart

- We can summarize overall objective of model-based design approach in the form of a flow chart

Now, we can summarize this overall objective of this model - based design approach in the form of a flowchart. So, essentially what we are talking of is a model based design approach. It has two types namely adaptive and non adaptive and two scopes namely automated and semi automated. Now, we can organize these types and scopes in the form of a flow chart as shown here.

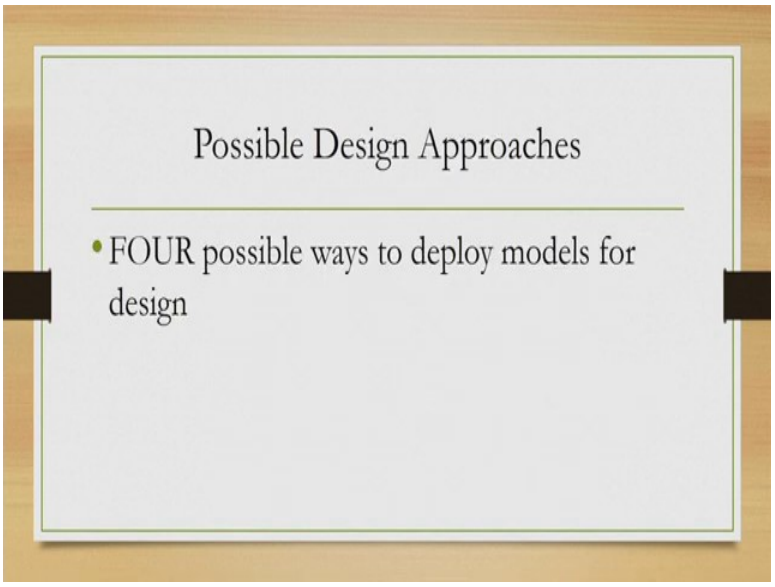
(Refer Slide Time: 09:46)



Objective Flow Chart

So, the overall objective is the model - based design, now these design we can perform for adaptive systems as well as non - adaptive systems and for each of these type of systems we can go for either semi automated design approach or an automated design approach. The last layer belongs to the scope and the intermediate layer belongs to the design type as we have already explained before.

(Refer Slide Time: 10:21)

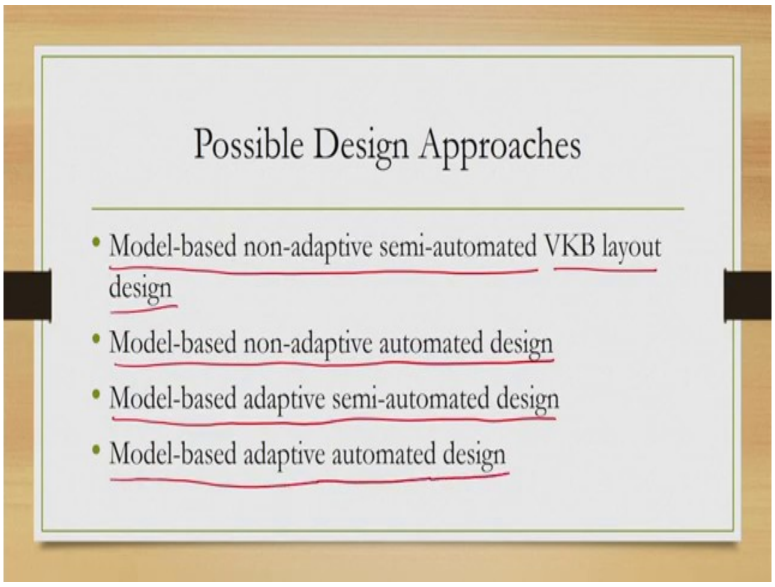


Possible Design Approaches

- FOUR possible ways to deploy models for design

So, going by this flowchart we can have four possible design approaches.

(Refer Slide Time: 10:27)



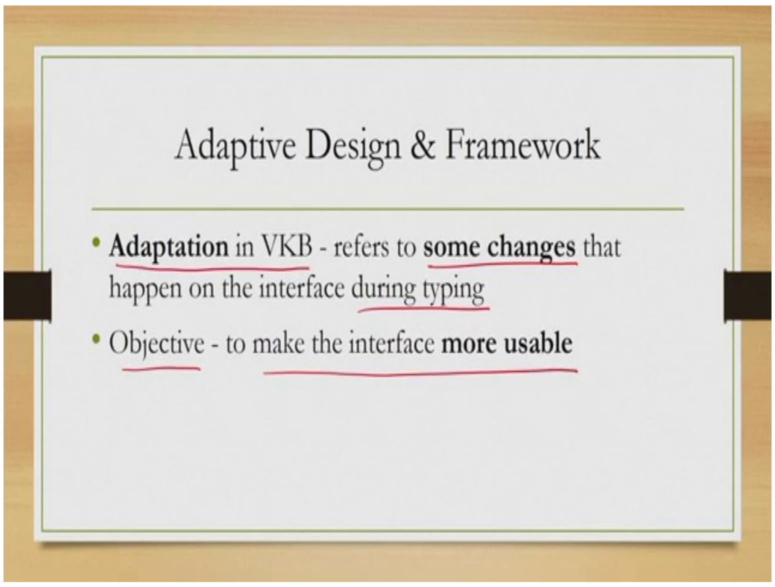
Possible Design Approaches

- Model-based non-adaptive semi-automated VKB layout design
- Model-based non-adaptive automated design
- Model-based adaptive semi-automated design
- Model-based adaptive automated design

One is model based non-adaptive semi - automated approach and in this context we are concerned about virtual keyboard design. So, model based non adaptive semi automated virtual keyboard layout design this is one design approach. Second approach is model based non adaptive automated virtual keyboard design. Third is model based adaptive semi automated virtual keyword design and finally, model based adaptive automated virtual keyboard design. So, these are the four ways using which we can use the models in order to design an interactive system.

Now, in the case of virtual keyboards we have chosen this as a case study to illustrate these design approaches that is the main objective of this lecture. However, in the case of virtual keyboard adaptive design creates some problem at this stage. So, it is difficult to actually explain adaptive design of virtual keyboards because of several reasons. Let us try to understand these reasons.

(Refer Slide Time: 11:41)

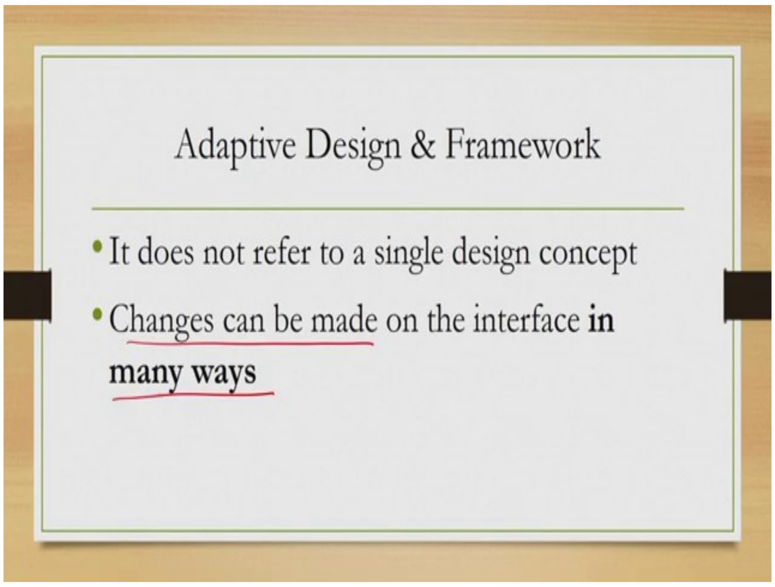


Adaptive Design & Framework

- **Adaptation** in VKB - refers to **some changes** that happen on the interface during typing
- Objective - to make the interface **more usable**

Now, when we talk of adaptive virtual keyboards or adaptation in virtual keywords it refers to some changes that happen on the interface during typing. So, we start with the design and during typing during use of that design something changes, that may be changing interface that may be changing interaction as we have defined adaptive interactive systems earlier. So, what is the objective, the overall objective is to make this interface more usable. So, by making those changes we assume that the interface becomes more usable.

(Refer Slide Time: 12:21)

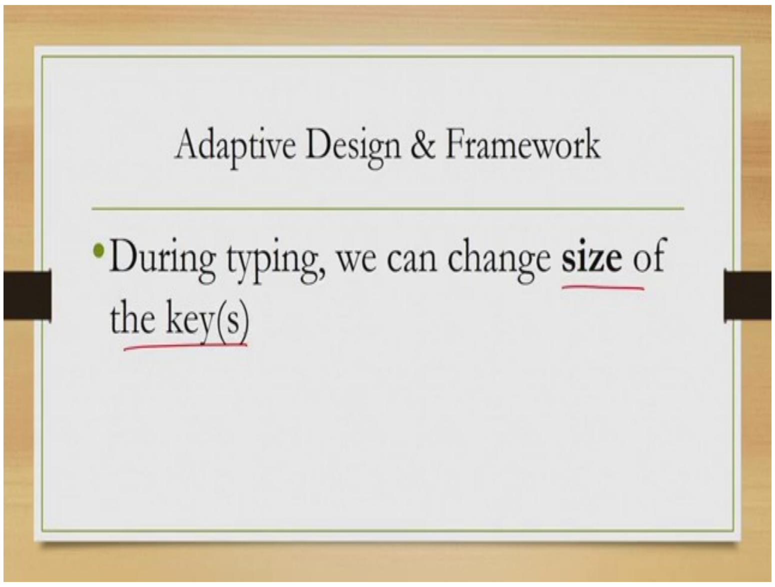


Adaptive Design & Framework

- It does not refer to a single design concept
- Changes can be made on the interface **in many ways**

However, these changes create the problem, because the changes that we make in the adaptive layouts of a virtual keyboard do not point to a single design concept. So, each change actually refers to one design concept and these changes can be made in many ways. So, for a layout it is difficult to define a specific design that reflect the adaptation. We can perform adaptation in many possible ways, what are those ways let us get some idea.

(Refer Slide Time: 13:08)

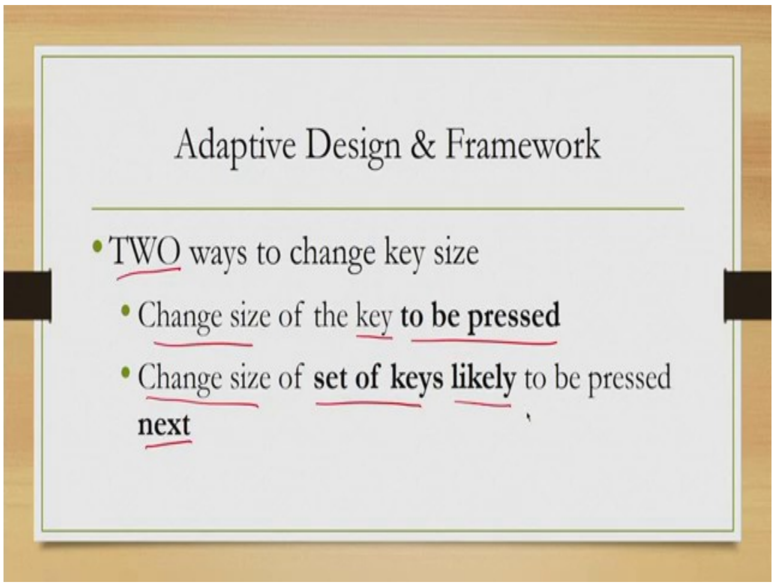


Adaptive Design & Framework

- During typing, we can change **size** of the key(s)

During typing we can change the size of the keys so that is one way to adapt. So, the size may become larger during typing. So, I want to type k, the size of the key k becomes large compared to the other keys in the keyboard. So, that is one form of adaptation.

(Refer Slide Time: 13:29)

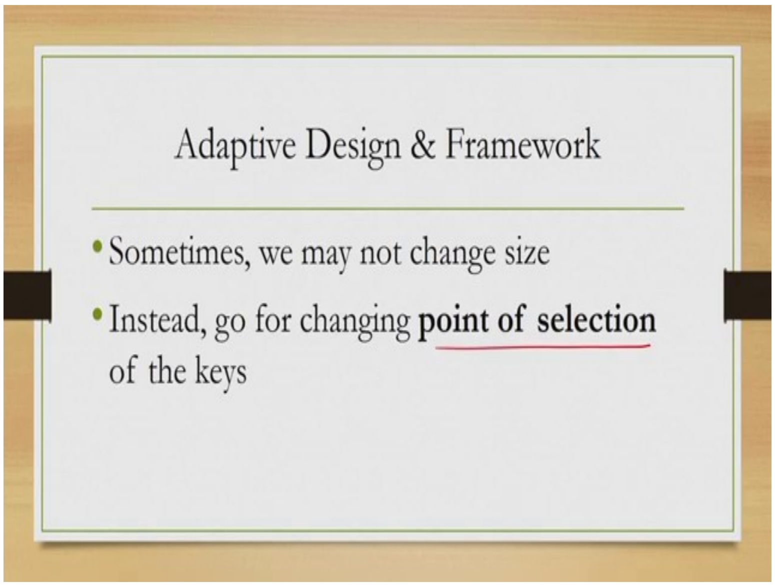


Adaptive Design & Framework

- TWO ways to change key size
  - Change size of the key **to be pressed**
  - Change size of **set of keys likely** to be pressed **next**

Alternatively what can happen is that we can change the set of keys that are likely to be placed next so, instead of changing the size of the current key. So, when I said that one strategy for adopting a layout is to change the size of key there are two possibilities. What are these two possibilities, one is we can change the size of the current key to be pressed or we can change the size of the set of keys that are likely to be pressed next after the current key press and each of these refer to one design.

(Refer Slide Time: 14:07)

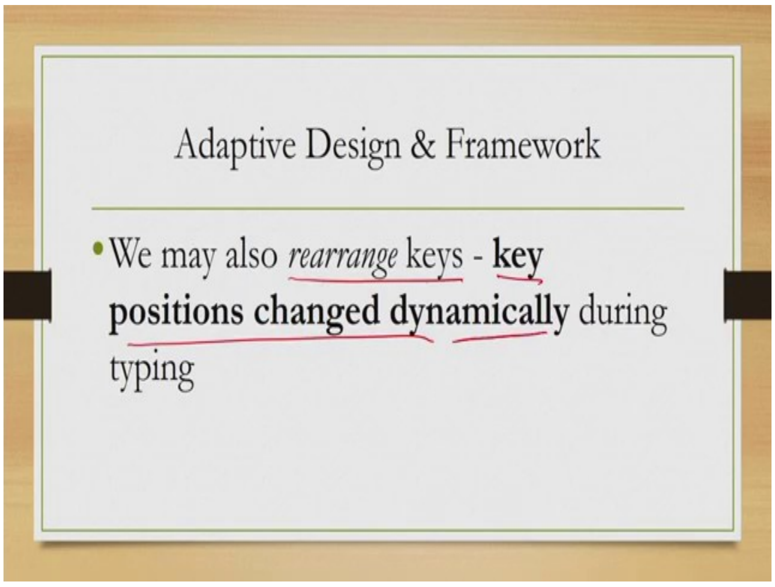


Adaptive Design & Framework

- Sometimes, we may not change size
- Instead, go for changing **point of selection** of the keys

Sometimes we may not change size instead we may change the point of selection. So, essentially the touch point which selects a particular key. So, that is kind of changing the interaction rather than the layout.

(Refer Slide Time: 14:24)

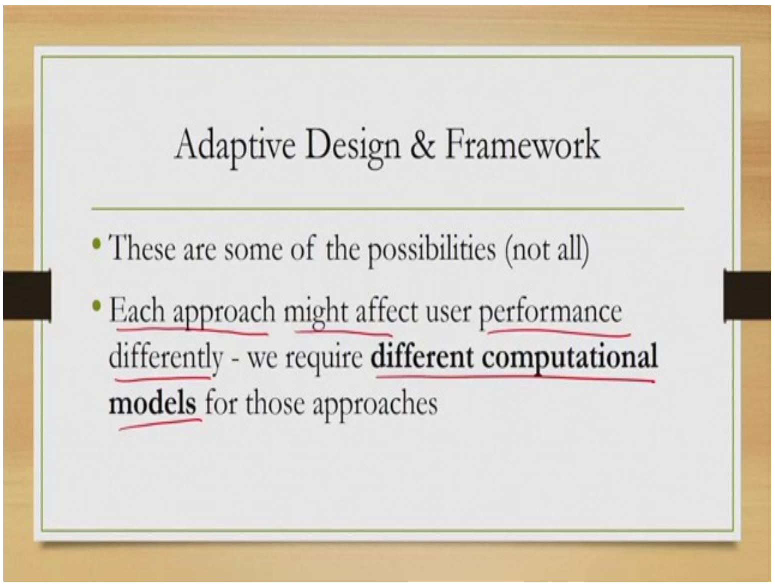


Adaptive Design & Framework

- We may also *rearrange* keys - **key positions changed dynamically** during typing

There is a third possibility also we can rearrange the keys, in other words the key positions change dynamically during typing. So, here we are not changing the size or touch point instead what we are changing is the position, that is the rearrangement of the keys on the layout.

(Refer Slide Time: 14:41)

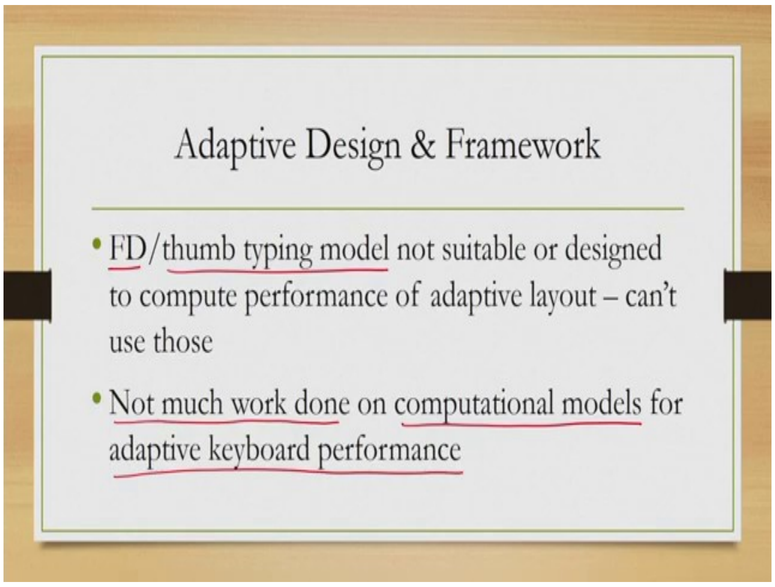


Adaptive Design & Framework

- These are some of the possibilities (not all)
- Each approach might affect user performance differently - we require **different computational models** for those approaches

And these are only some of all the possibilities the ways we can adapt layout the problem is each of these refer to particular design and for that design we need to have a model. So, each approach might affect the performance differently and in order to predict that difference we require different computational models.

(Refer Slide Time: 15:15)



Adaptive Design & Framework

- FD/thumb typing model not suitable or designed to compute performance of adaptive layout – can't use those
- Not much work done on computational models for adaptive keyboard performance

Now, if you can recollect earlier we have discussed two mobile typing models, the Fitts diagraph model and the thumb typing model. These models are not designed for modeling performance in the context of adaptive keyboards. So, we cannot apply them to model performance of a user who is using an adaptive virtual keyword layout. And unfortunately not much work has been done on the computational models for adaptive keyboard typing performance.

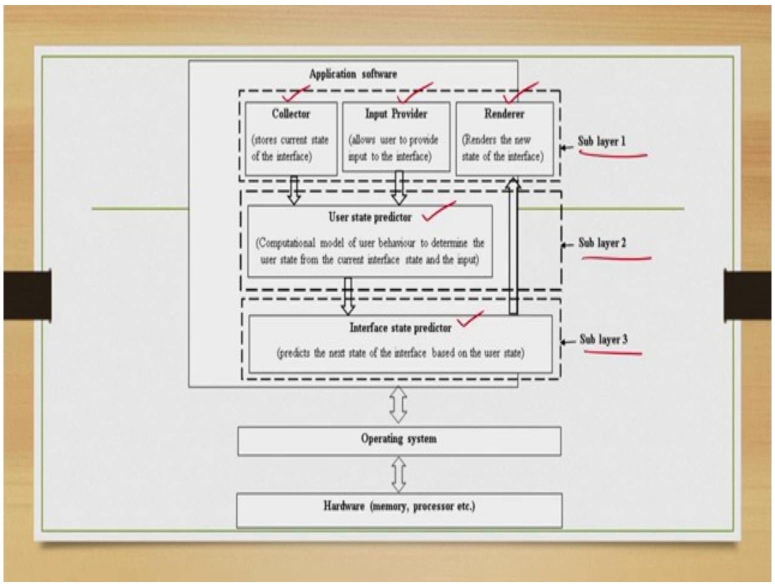
So, what it tells is that we do not have models and since we do not have models it will be difficult to illustrate the use of models in the context of adaptive keyboard design. So, among the four possible design approaches, the two approaches that are related to adaptive design namely semi automated adaptive layout design and automated adaptive layout design are not possible to be discussed because we do not have the suitable models. We will focus only on the design of non adaptive keyboard layouts.

(Refer Slide Time: 16:28)

Non-adaptive semi-automated design

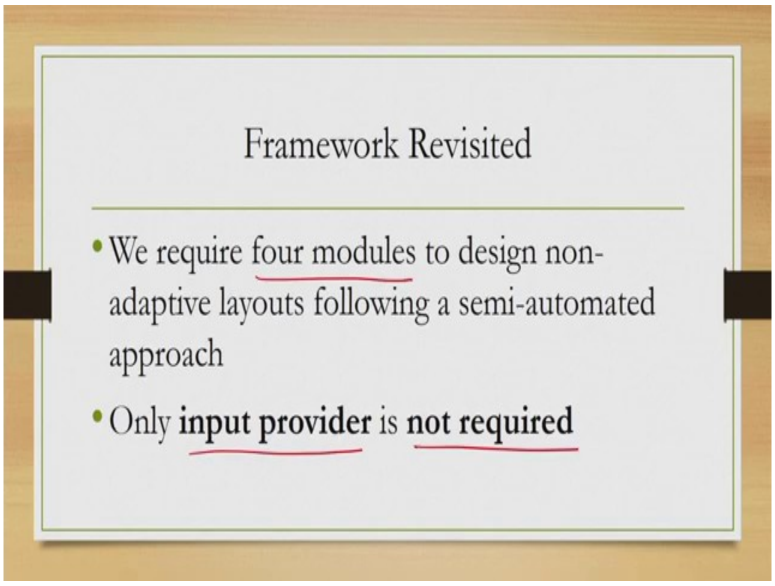
So, let us start with the non adaptive semi automated design approach. So, how we can use models in order to develop virtual keyboard layout that is non adaptive, where the design approach followed is semi automated. Our objective is to show these approach in light of the computational framework that we have proposed earlier, how to utilize the idea of the computational framework to illustrate the design of non adaptive virtual keyboard layout using a semi automated approach.

(Refer Slide Time: 17:04)


Application software
Collector
(stores current state of the interface)
Input Provider
(allows user to provide input to the interface)
Renderer
(Renders the new state of the interface)
Sub layer 1
User state predictor
(Computational model of user behaviour to determine the user state from the current interface state and the input)
Sub layer 2
Interface state predictor
(predicts the next state of the interface based on the user state)
Sub layer 3
Operating system
Hardware (memory, processor etc.)

So, if you can recollect we have in the framework five modules. So, in the framework we have five modules, 3 of the modules are organized in sub layer 1 namely the collector, the input provider and the renderer. One module is there in sub layer 2 that is the user state predictor. And in sub layer 3 there is the 5th module the interface state predictor that is our framework.

(Refer Slide Time: 17:35)

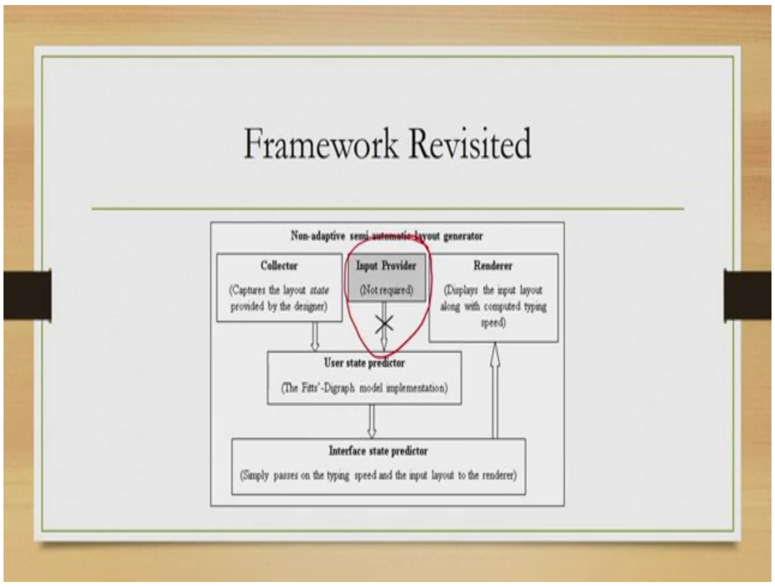


Framework Revisited

- We require four modules to design non-adaptive layouts following a semi-automated approach
- Only **input provider** is **not required**

Now, to explain the idea of design of non adaptive virtual keyboard layout following semi automated approach, we do not require all these five modules. Instead what we required, are four modules, the input provider module is not required. So, the first thing we should do is modify our framework for explaining this particular situation.

(Refer Slide Time: 18:04)

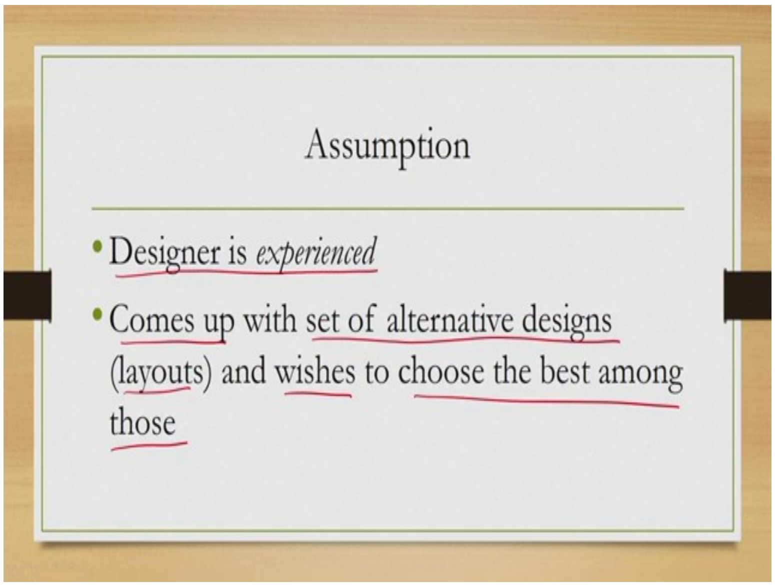


Framework Revisited

Non-adaptive semi automatic layout generator
Collector
(Captures the layout *state* provided by the designer)
Input Provider
(Not required)
Renderer
(Displays the input layout along with computed typing speed)
User state predictor
(The Fitts'-Digraph model implementation)
Interface state predictor
(Simply passes on the typing speed and the input layout to the renderer)

So, here we have shown it with a cross in the connection. So, the input provider module is not required we are living it out and the remaining modules stay in the framework which explains the, implementation of semi automated design approach for non adaptive virtual keyword layout design.

(Refer Slide Time: 18:29)



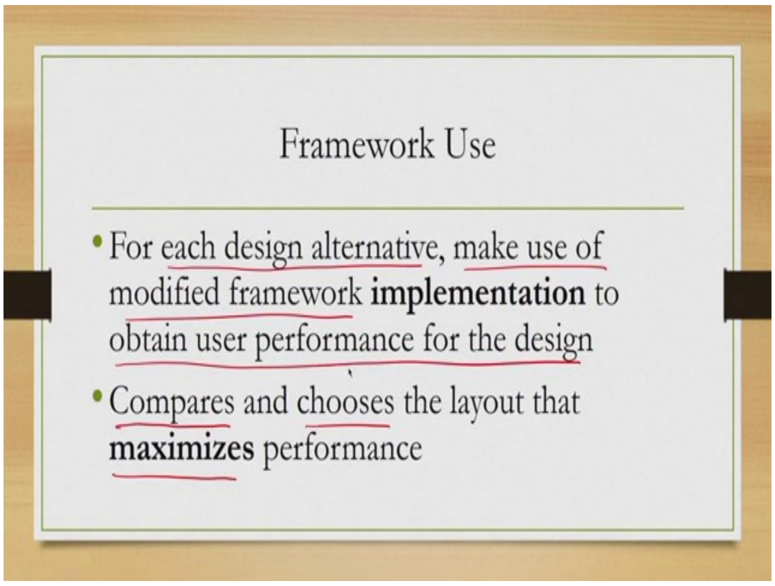
Assumption

- Designer is *experienced*
- Comes up with set of alternative designs (layouts) and wishes to choose the best among those

Before we go for further explanation how the four modules are implemented what we should do in those modules to implement them. We start with the assumptions that we make in order to use the framework to implement the design approach. The first assumption is that the designer is experienced. So, the designer already has some experience of designing similar systems that is the first assumption designer is experienced.

The second assumption is based on the experience the designer can come up with a set of alternative designs or the layouts and wishes to choose the best among those. So, two things you should note, one is the designer is capable of coming up with alternative designs and second thing is that among those alternatives the designer wishes to choose by himself or herself the best one.

(Refer Slide Time: 19:37)



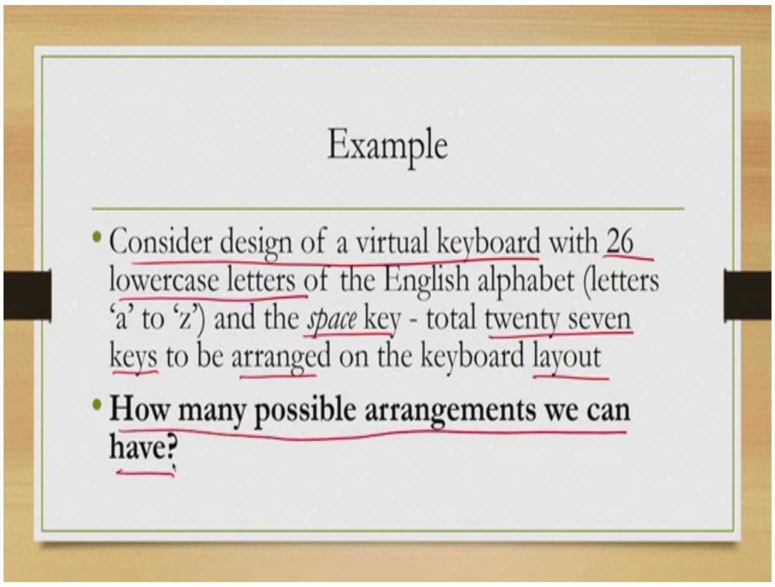
Framework Use

- For each design alternative, make use of modified framework **implementation** to obtain user performance for the design
- Compares and chooses the layout that **maximizes** performance

So, then how the framework can be implemented to design non adaptive virtual keyboard layout let us see. So, we have already assumed that the designer comes up with alternatives, for each design alternative the designer makes use of the framework which you have modified by removing the input provider component, rather the designer make use of the implementation of the framework to obtain user performance for the design and based on the performance reported by the framework implementation. Compares and chooses the best layout, which is the best layout that maximizes the performance.

In the case of virtual keyboard the performance is represented by the text entry rate so, whichever layout maximizes text entry rate is chosen by the designer. Now, let us use some specific keyboard examples to understand the concept better.

(Refer Slide Time: 20:53)

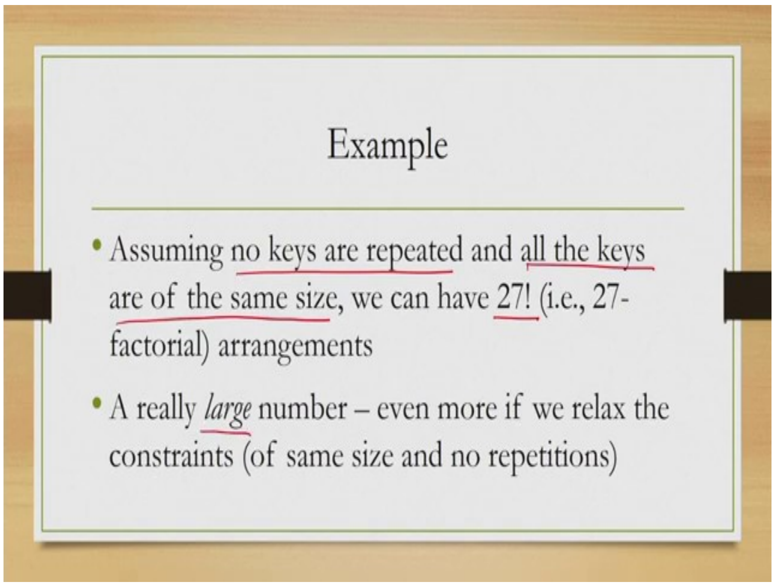


Example

- Consider design of a virtual keyboard with 26 lowercase letters of the English alphabet (letters 'a' to 'z') and the *space* key - total twenty seven keys to be arranged on the keyboard layout
- **How many possible arrangements we can have?**

Consider the design of a keyboard which has only 26 lower case letters. So, we are excluding capital letters and other punctuation marks and numeric characters. So, along with these 26 lowercase letters the keyboard also has this space key. So, there are there is a total of 27 keys to be arranged on the layout. So how many layouts are possible with 27 keys? In how many different ways we can arrange these 27 keys?

(Refer Slide Time: 21:37)

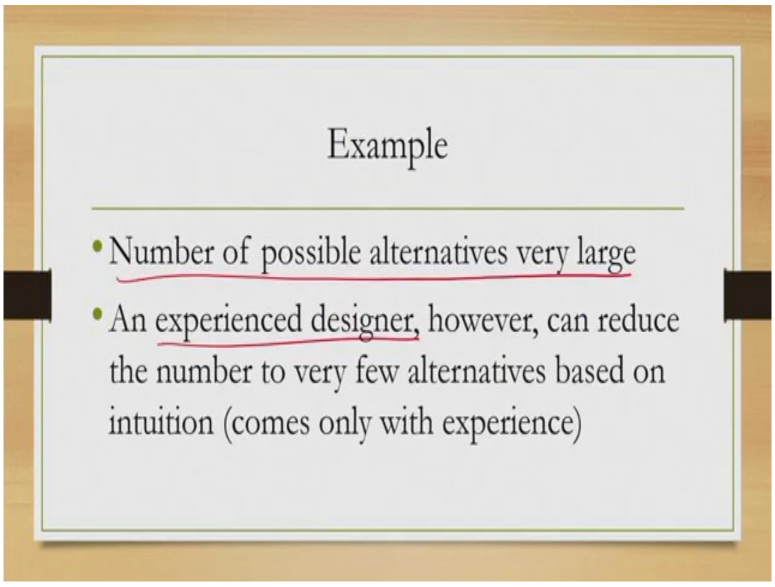


Example

- Assuming no keys are repeated and all the keys are of the same size, we can have 27! (i.e., 27-factorial) arrangements
- A really *large* number – even more if we relax the constraints (of same size and no repetitions)

If you think then you can come up with a figure like, 27 factorial, where we can assume that no keys are repeated and all the keys are of the same size. If we relax this these constraints then the number will be even larger, but in any case even if we have these constraints. So, we can see that the 27 factorial the number of possible layouts is a very large number in itself.

(Refer Slide Time: 22:10)

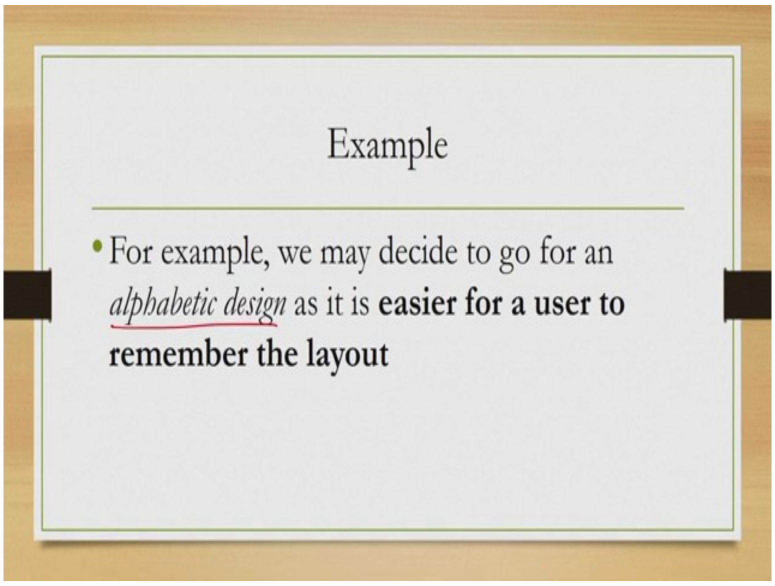


Example

- Number of possible alternatives very large
- An experienced designer, however, can reduce the number to very few alternatives based on intuition (comes only with experience)

So, the number of possible alternatives is very large. So, all the layouts in this set of factorial 27 number of layouts are possible candidates for a design and we can safely say that this number is very large number for practical consideration. So, what is the advantage that an experienced designer has? Essentially, based on experience and intuition a designer can come up with very few alternatives and experienced designer need not check this large number of layouts to come up with appropriate alternatives. Instead experience and intuition helps to come up with less number of alternatives that are manageable.

(Refer Slide Time: 23:11)



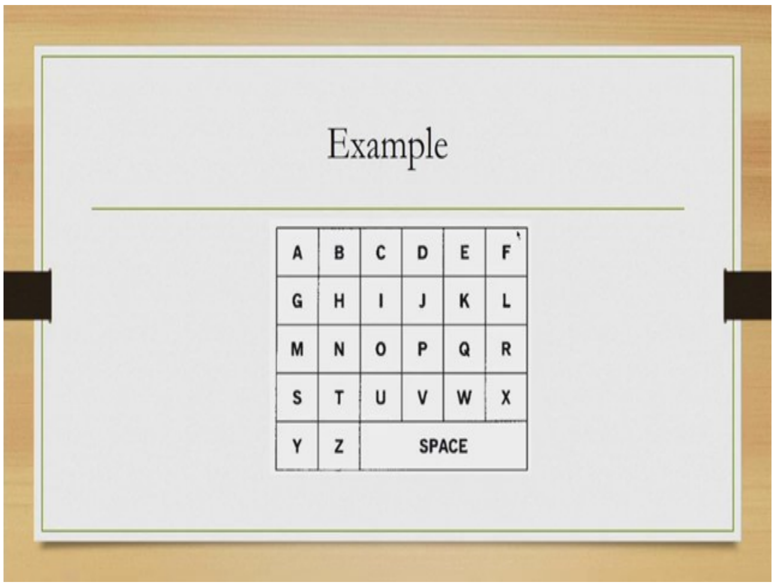
Example

- For example, we may decide to go for an *alphabetic design* as it is **easier for a user to remember the layout**

As an example, we can think of two alternatives which intuitively tells us that the corresponding designs are likely to be usable. The first alternative is an alphabetic design, what an alphabetic design means; it means that the letters on the layout are arranged alphabetically the way they are arranged in the alphabet and why this design is going to be good, because it is easier to remember the position of the keys.

So, user can gain familiarity with this layout quickly, because we are all familiar with the alphabet and we know that the keys are arranged alphabetically. So, we know where to look for a particular key in the layout.

(Refer Slide Time: 24:00)

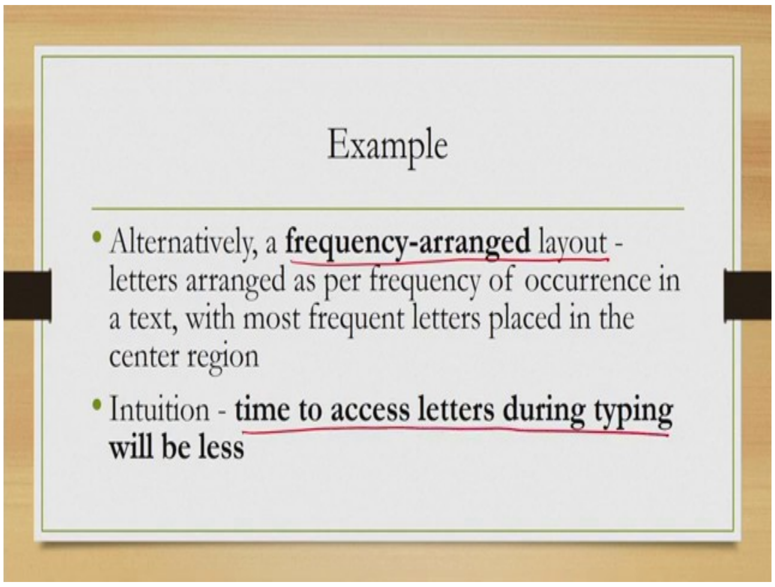


Example

A	B	C	D	E	F
G	H	I	J	K	L
M	N	O	P	Q	R
S	T	U	V	W	X
Y	Z	SPACE			

The keyboard design may look like this is of course, one possibility. So, here as you can see the keys are arranged alphabetically A is followed by B followed by C followed by D the way they are included in the alphabet and it helps in learnability. We can learn the layout quickly and also memorability we can remember the position of the keys easily.

(Refer Slide Time: 24:36)



Example

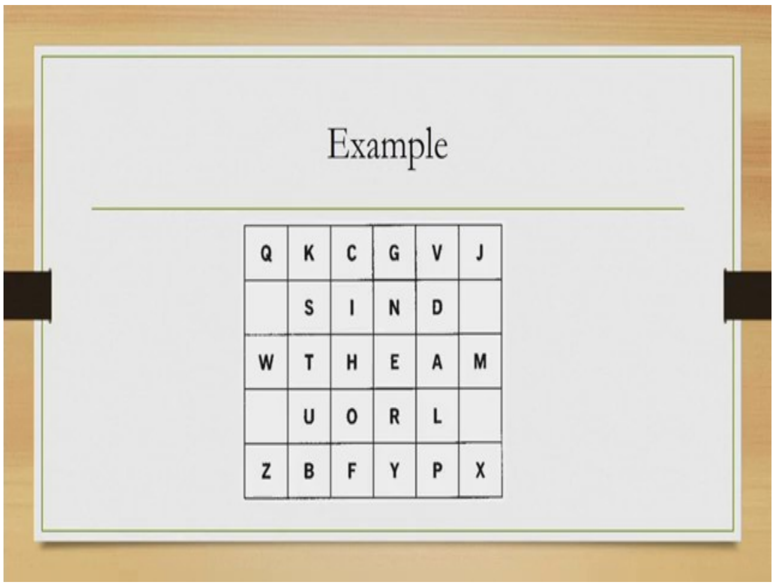
- Alternatively, a **frequency-arranged** layout - letters arranged as per frequency of occurrence in a text, with most frequent letters placed in the center region
- Intuition - **time to access letters during typing will be less**

Now, let us think of an alternative design that is a frequency arranged layout. So, what this layout means, it means that we find out the frequency of occurrence of each letter in text based on these frequencies we arrange them on the layout. Typically, the letters or the characters that occur more frequently are placed towards the central region of the layout and the more the character is away from the central region, the less frequent it is in text. Why central region? Because it has been found that the central region draws our attention before other regions. So, we prefer to keep them in the central regions.

One question is, how to get these frequencies? We can use a language corpus remember that corpus is a collection of texts where the texts were collected following a systematic and scientific approach and for a language we have corpus, which contains texts of that language and using this corpus we can find out unigram probability distributions of the alphabetic letters in those texts and that unigram distribution can give us the relative importance of characters with respect to other characters in terms of frequency.

Now, why frequency arranged layout is likely to be a good choice because it reduces the time to access letters during typing. So, essentially we are trying to reduce the typing time and increase the typing speed and frequency arranged layouts are likely to help us increasing typing speed. So, let us see one example of a frequency arranged layout with this 27 keys.

(Refer Slide Time: 27:00)

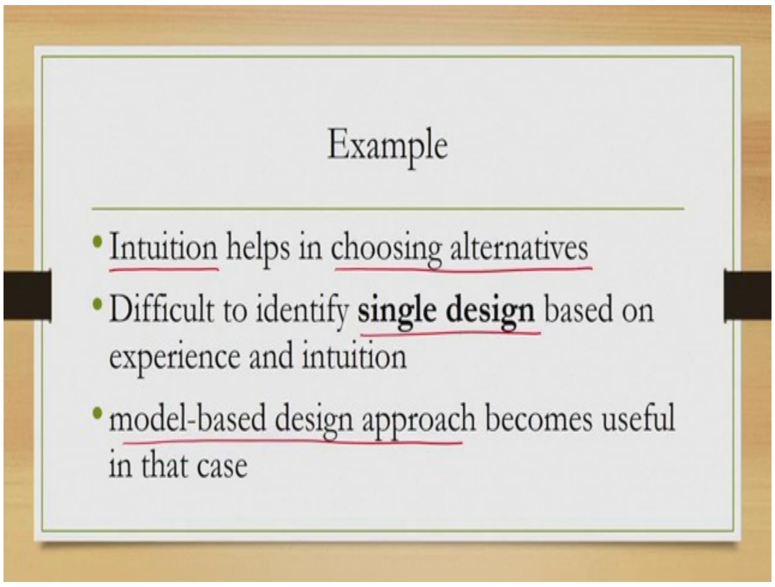


Example

Q	K	C	G	V	J
	S	I	N	D	
W	T	H	E	A	M
	U	O	R	L	
Z	B	F	Y	P	X

So, the layout that is shown here shows one possibility and the arrangement is based on the unigram distribution of English letters obtained from a written corpus of English language.

(Refer Slide Time: 27:21)

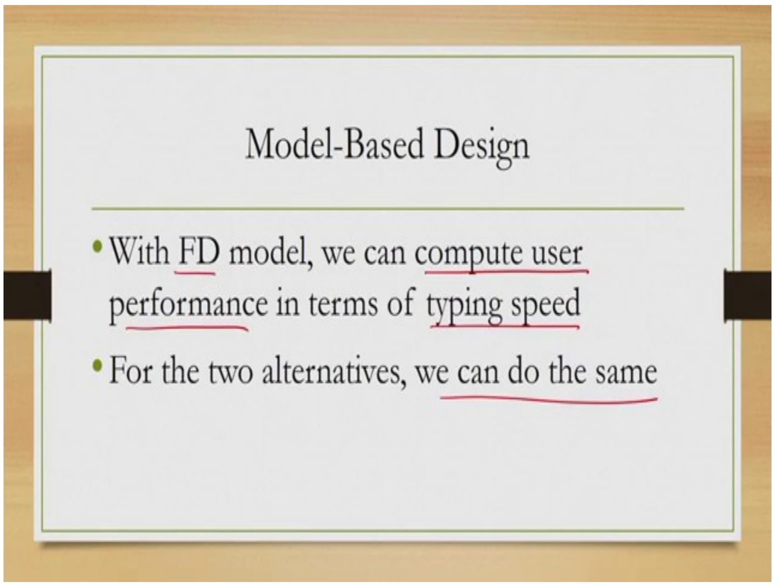


Example

- Intuition helps in choosing alternatives
- Difficult to identify **single design** based on experience and intuition
- model-based design approach becomes useful in that case

Now, with the intuition with the intuition of the designer based on experience of course, the designer can choose these two alternatives. The problem is which one among them will be the best. Just by looking at the layout it is of course, difficult to identify the single design among these alternatives in this case the two designs which one will be the best and they are this model based approach becomes useful. Let us see how, let us see how we can implement the framework so that we can choose the best among the two possible alternatives or a set of possible alternatives using the models.

(Refer Slide Time: 28:15)

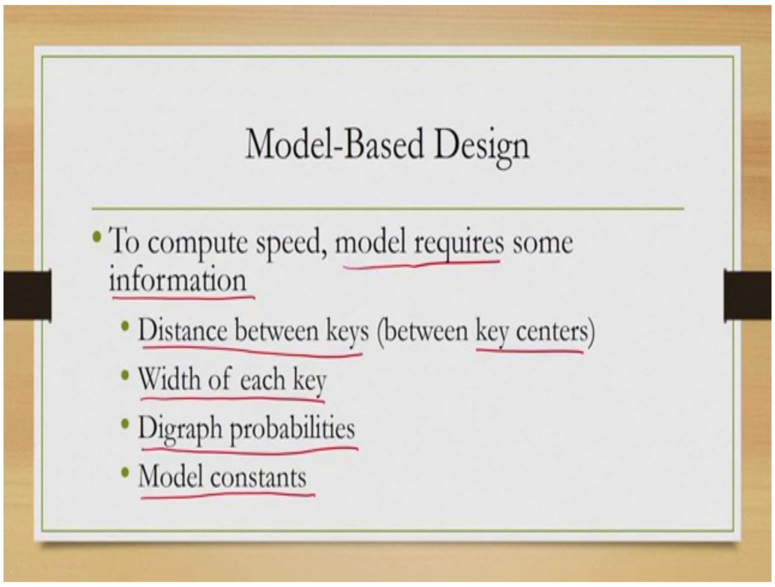


Model-Based Design

- With FD model, we can compute user performance in terms of typing speed
- For the two alternatives, we can do the same

Recollect that with the Fitts diagraph model which is a model for single finger typing, we can compute performance in terms of typing rate. Now for the two alternatives we can do the same for each alternative we apply the Fitts diagraph model to compute performance or the typing speed and the one that gives the better or the higher typing speed will be the one that we should choose.

(Refer Slide Time: 28:43)

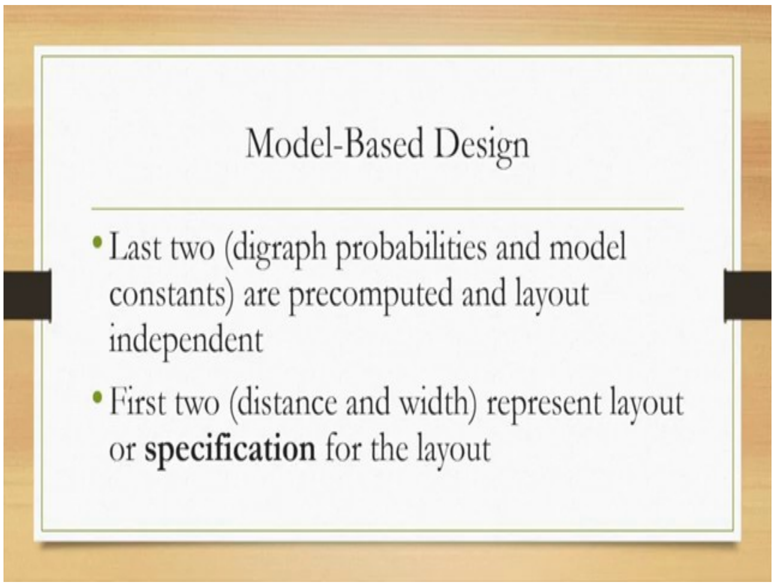


Model-Based Design

- To compute speed, model requires some information
  - Distance between keys (between key centers)
  - Width of each key
  - Digraph probabilities
  - Model constants

Now in order to compute the speed the model requires some information, what are those information. The distance between keys or between the key centers because we are assuming rectangular keys, width of each key, diagraph or bigram probabilities which we can get from a corpus and the model constants, that is the constants A and B in the Fitts law component of the model or the Hick Hyman law component of the model.

(Refer Slide Time: 29:21)

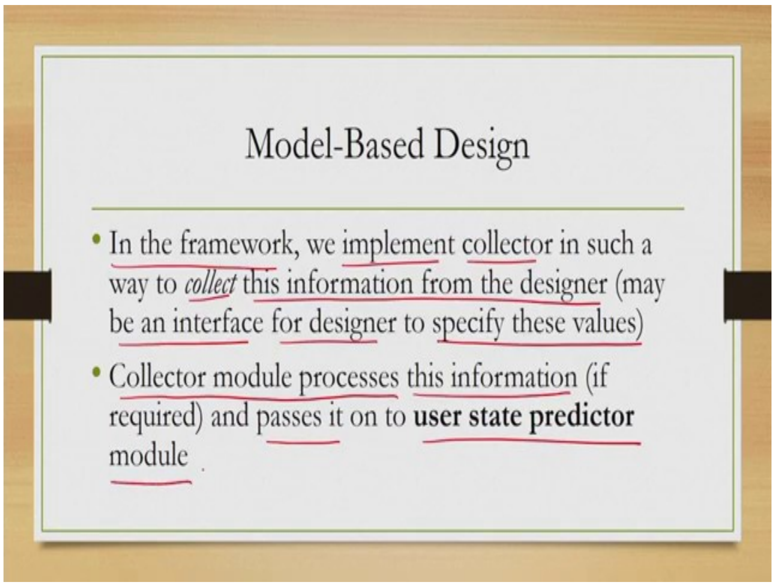


Model-Based Design

- Last two (digraph probabilities and model constants) are precomputed and layout independent
- First two (distance and width) represent layout or **specification** for the layout

Now, the digraph probabilities and the model constants are pre computed and they are independent of the layout, but the first two the distance between keys and the width of each key represents the layout or specification for the layout. So, they are very much layout dependent. So, then what we can do.

(Refer Slide Time: 29:50)



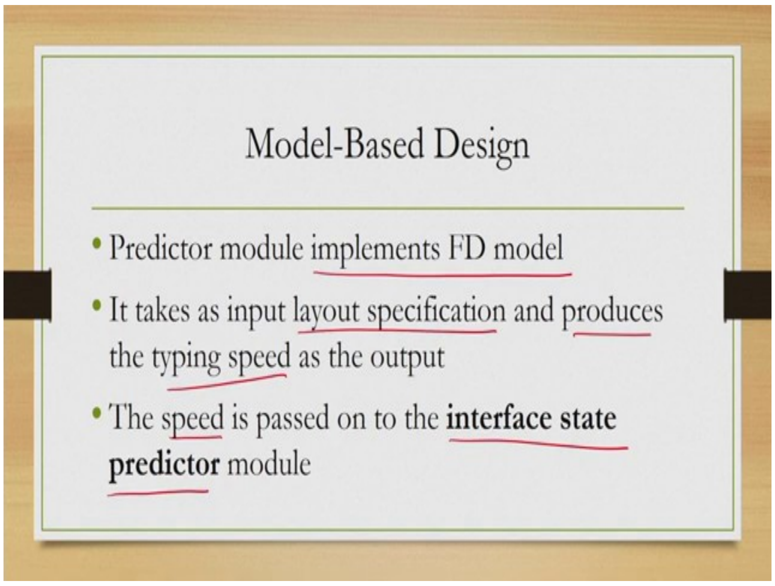
Model-Based Design

- In the framework, we implement collector in such a way to *collect* this information from the designer (may be an interface for designer to specify these values)
- Collector module processes this information (if required) and passes it on to **user state predictor** module

So, in the framework, we implement the collector in such a way such that we can collect this information from the designer which information, the distance between keys and the key width, width of each key. It may be implemented by providing an interface to the designer for specifying these values that is the implementation of the collector module.

Now, collector module processes this information if required; that means, if there is some conversion required or some rounding off required and so on and passes on this information to the user state predictor model, remember that we do not have any input provider module in this case.

(Refer Slide Time: 30:46)

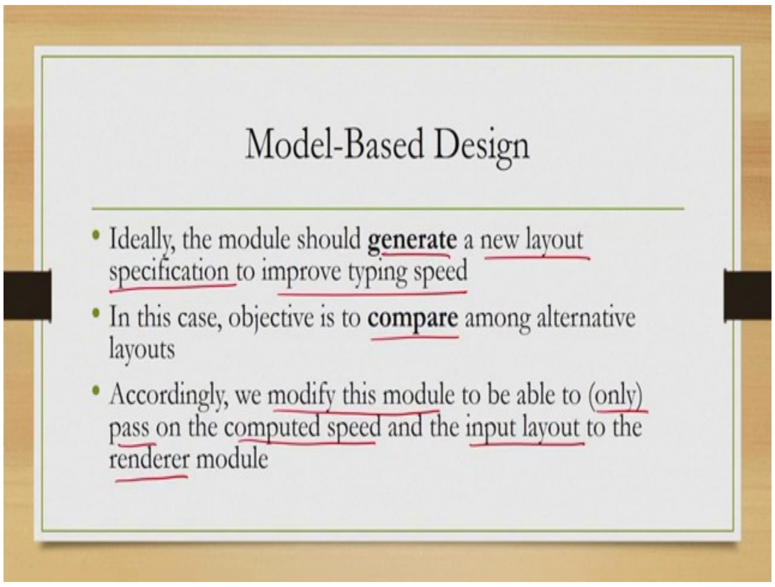


Model-Based Design

- Predictor module implements FD model
- It takes as input layout specification and produces the typing speed as the output
- The speed is passed on to the **interface state predictor** module

Now, the predictor module implements the Fitts diagraph model. So, in this implementation of the framework the predictor module is an implementation of the Fitts diagraph model and it takes as input the layout specific and produces the typing speed as output. The speed is passed on to the interface state predictor module.

(Refer Slide Time: 31:16)



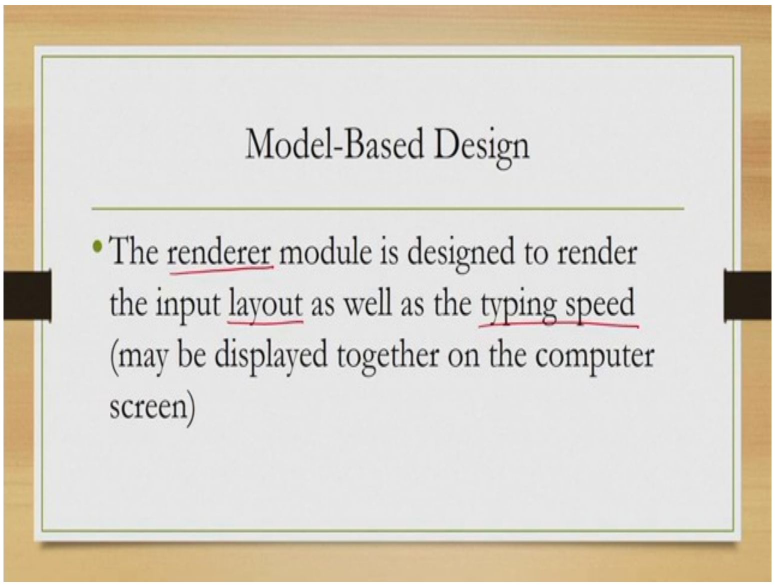
Model-Based Design

- Ideally, the module should **generate** a new layout specification to improve typing speed
- In this case, objective is to **compare** among alternative layouts
- Accordingly, we modify this module to be able to (only) pass on the computed speed and the input layout to the renderer module

In a very generalized situation this interface state predictor module should generate a new layout or the specification of the layout to improve the typing speed. However, in this case our objective is to compare rather than generate final layout. So, here our interface state predictor module can be implemented simply. So, in this case what we can do is basically we can modify this module. So, that the module is able to only pass on the computed speed and the input layout as is to the renderer module.

So, here we do not need to generate new specification or specification for a new layout instead we simply collect this speed information computed speed information from the user state predictor module and the input layout specification and pass on these two information to the renderer module. So, this is a very simple implementation of the interface state predictor module. And what is the job of the renderer module?

(Refer Slide Time: 32:39)



Model-Based Design

- The renderer module is designed to render the input layout as well as the typing speed (may be displayed together on the computer screen)

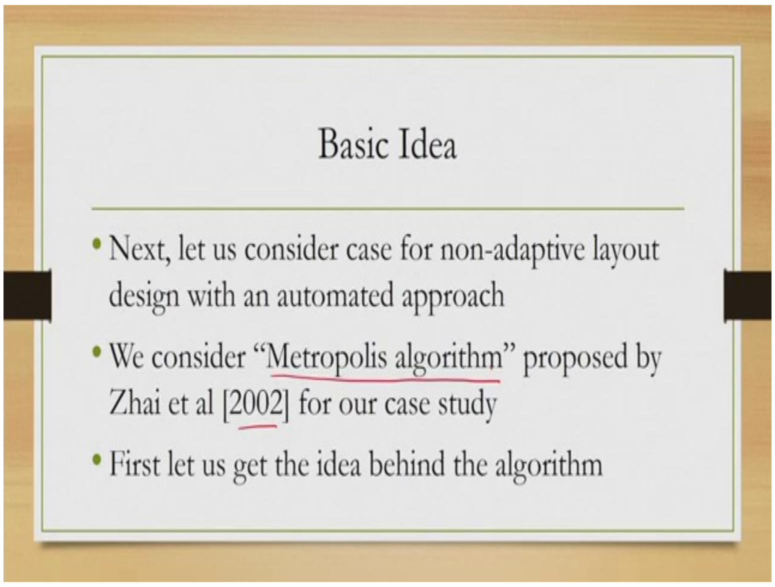
So, it simply needs to render the computed speed to the user. So, along with that it also renders the layout which is already specified. So, essentially the renderer modules should be implemented in a way such that it is capable of displaying the layout and display the typing speed on the screen that is the implementation. So, to summarize in order to implement the framework to develop non adaptive layout which is designed using semi automated approach, what we can do?

We can rely on the designer to come up with alternative designs and for each design we use the framework to compute the speed and based on the speed that the framework outputs the designer makes a choice. Now, how the frame work can display the speed or how the framework can output the speed information, which is a major of typing performance? So, essentially the input provider module is not there. So, the collector module should be implemented in a way such that the designer is capable of specifying a layout in terms of distance between keys and the key width.

This specification is passed on to the user state predictor model, the user state predictor module in this case it is a simple implementation of the Fitts diagraph model or it may implement the thumb typing model depending on which kind of behavior we want to predict. Now, this model computes the speed based on the specification and pass it on to interface state predictor. The interface state predictor in turn passes on the speed as well as the input specification to the renderer module.

So, the input specification also comes to the interface state predictor through the user state module and both these are passed onto the renderer module. So, renderer module displays the layout the specified layout along with the typing speed computed with the model. And these are noted down by the designer. Once it is done for all the possible alternatives the designer can compare the typing speeds and choose the one that gives the best typing speed. Now, let us discuss how we can implement the framework for automated non adaptive keyboard design.

(Refer Slide Time: 35:38)

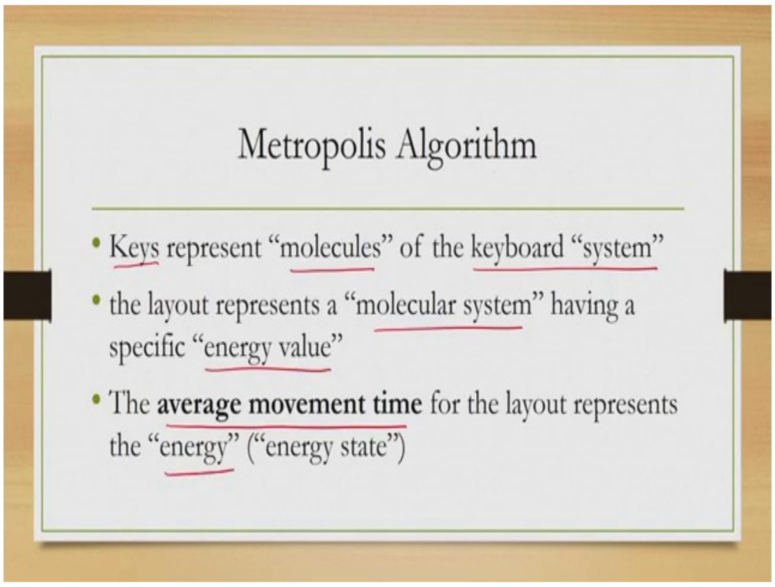


Basic Idea

- Next, let us consider case for non-adaptive layout design with an automated approach
- We consider "Metropolis algorithm" proposed by Zhai et al [2002] for our case study
- First let us get the idea behind the algorithm

So, in order to understand this automated design approach we will make use of one algorithm for automatic design of virtual keyboards that was proposed by Zhai et al in 2002, this is one "metropolis algorithm." Now before discussing the implementation of the framework, first let us try to get the idea behind the algorithm.

(Refer Slide Time: 36:06)



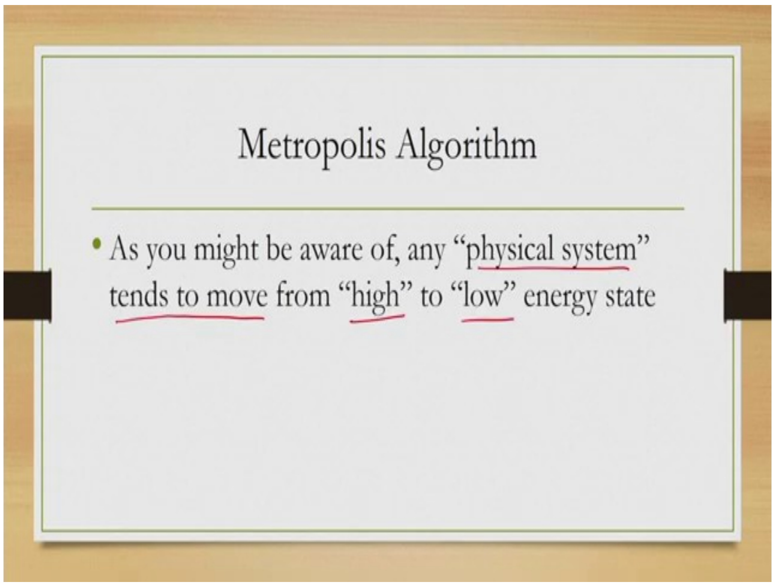
Metropolis Algorithm

- Keys represent "molecules" of the keyboard "system"
- the layout represents a "molecular system" having a specific "energy value"
- The **average movement time** for the layout represents the "energy" ("energy state")

What this algorithm states, it states that the keyboard can be considered as a system of "molecules", where the keys are the molecules and together they represent the keyboard "system" which is a molecular system as we find in nature. So, the overall layout represents a "molecular system" having a specific "energy value" which is again a natural thing.

What is the energy value in case of our keyboard system, it is defined as the average movement time for a layout. So, the average movement time for the layout is the "energy". Now a keyboard with this specific average movement time is said to be in the specific energy state.

(Refer Slide Time: 37:00)

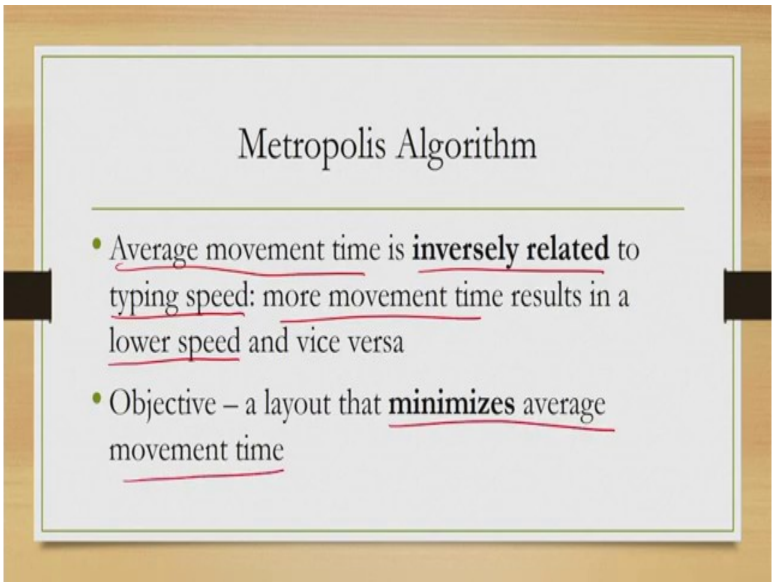


Metropolis Algorithm

- As you might be aware of, any "physical system" tends to move from "high" to "low" energy state

And you may be aware that what happens in nature is, any "physical system" tends to move from "high" energy state to "low" energy state. So, that is the natural tendency of any physical system, here we are equating keyboard to a physical system. So, in this case also our objective is to take the keyboard from a high energy state to a low energy state.

(Refer Slide Time: 37:35)



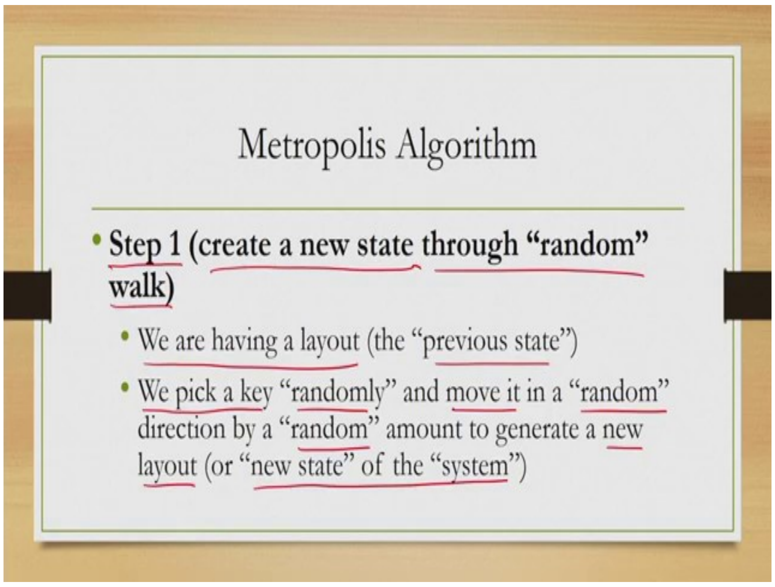
Metropolis Algorithm

- Average movement time is **inversely related** to typing speed: more movement time results in a lower speed and vice versa
- Objective – a layout that **minimizes** average movement time

So essentially what the algorithm assumes that, we have a keyboard layout which is having a specific energy state and the algorithm tries to find out another layout which has a lesser energy state essentially from a higher energy state the algorithm tries to take the keyboard system to a lower energy state and the way energy is defined it indicates that the keyboard with lower energy state will have lower movement time.

Now, as we know the average movement time is inversely related to typing speed: so, the more the movement time the lower is the speed and vice versa. Now, our objective is to take to a layout that minimizes the average movement time so that the typing speed maximizes. And minimization of the average movement time indicates that the objective of the algorithm is to take the keyboard system to a energy state having the lowest energy.

(Refer Slide Time: 38:43)



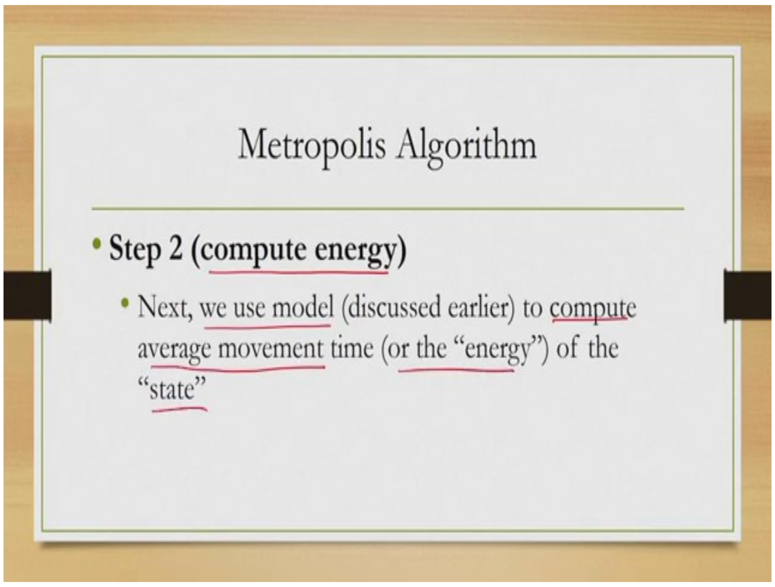
Metropolis Algorithm

- Step 1 (create a new state through "random" walk)
  - We are having a layout (the "previous state")
  - We pick a key "randomly" and move it in a "random" direction by a "random" amount to generate a new layout (or "new state" of the "system")

So what it does several steps. So, there are few steps in the algorithm in step 1 what we do is, we create a new state of the system through a "random" walk. So, we are having a layout at the beginning the "previous state" and we pick a key "randomly" and move it in a "random" direction by a "random" amount to generate a new layout or "new state" of the "system".

So, each layout corresponds to a state. So, what we do is, essentially from the current layout we pick up a key randomly and move it in a random direction by random amount to generate a new layout or a new energy state.

(Refer Slide Time: 39:36)

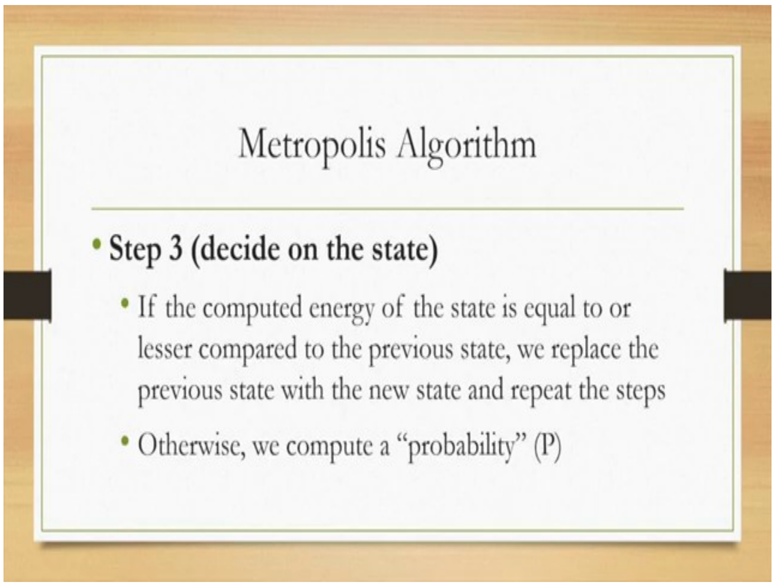


Metropolis Algorithm

- **Step 2 (compute energy)**
  - Next, we use model (discussed earlier) to compute average movement time (or the "energy") of the "state"

Then we compute the energy of the new state and note that we already have the energy of the previous state, how to compute the energy. We use the model to compute the average movement time. If you recollect our discussion during the Fitts digraph model so, before we computed the typing speed we first computed average movement time. So, up to that point whatever we have discussed we used that model to compute the average movement time which is essentially the energy of the new state.

(Refer Slide Time: 40:17)



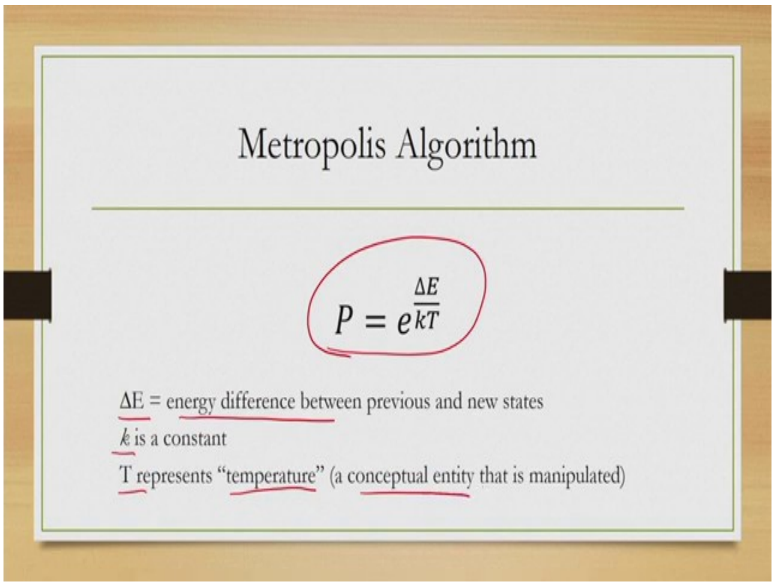
Metropolis Algorithm

- **Step 3 (decide on the state)**
  - If the computed energy of the state is equal to or lesser compared to the previous state, we replace the previous state with the new state and repeat the steps
  - Otherwise, we compute a "probability" (P)

And in step 3 we decide on what to do with the computed new state. If the computed energy of the new state is equal to or lesser compared to the previous state, we replace the previous state with the new state, because we are moving towards lower energy state and repeat the steps. Now, what happens when the new state is having a higher energy than the previous state, should we keep the previous state or not? The answer is not very straight forward, because there is a concept called local minima.

So it may happen that during the computation the overall system got struck in a local minima and there even if some change happens, that may not reflect that the system is moving or not moving towards a lower energy state. So, what we do there is, we compute "a probability", probability of retaining the new state or discarding the new state.

(Refer Slide Time: 41:46)



Metropolis Algorithm

$$\boxed{P = e^{\frac{\Delta E}{kT}}}$$

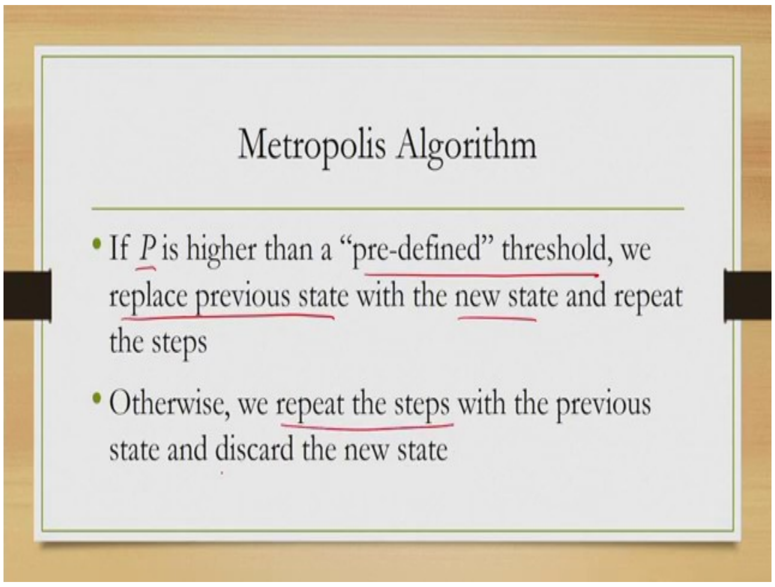
ΔE = energy difference between previous and new states

k is a constant

T represents "temperature" (a conceptual entity that is manipulated)

So, that probability is computed with this expression, where delta E represents the energy difference between the two states, k is a constant and T represents something equivalent to "temperature" in a physical system, which is a conceptual entity that is manipulated by the designer of the algorithm or the implementer of the algorithm.

(Refer Slide Time: 42:15)



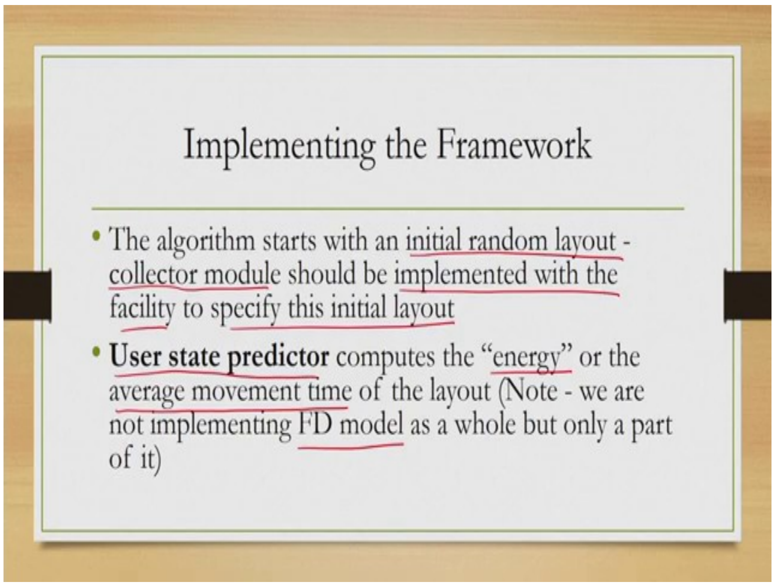
Metropolis Algorithm

- If P is higher than a "pre-defined" threshold, we replace previous state with the new state and repeat the steps
- Otherwise, we repeat the steps with the previous state and discard the new state

Now, once we compute P we compare that P value with a "pre-defined" threshold, now if the computed P is higher than the threshold then we replace the previous state with the new state and repeat the step. In case P is less than the predefined threshold then we do not replace the state we keep the previous state and repeat the steps so, in that case we actually discard the new state. So, we will not actually go into the details of why this is so, why we perform all these steps, what is the reason behind this.

If you are interested you can refer to the reference material that will be mentioned at the end of this lecture, the original work as well as the explanation can be found in the reference materials. Now, let us see with this basic understanding of the algorithm, let us see how we can implement the framework.

(Refer Slide Time: 43:18)



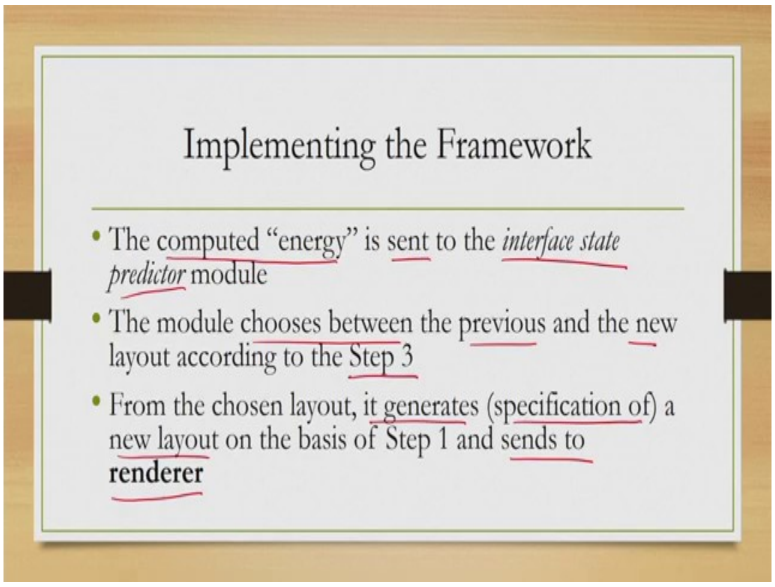
Implementing the Framework

- The algorithm starts with an initial random layout - collector module should be implemented with the facility to specify this initial layout
- **User state predictor** computes the "energy" or the average movement time of the layout (Note - we are not implementing FD model as a whole but only a part of it)

So, the algorithm starts with an initial random layout; that means, initially we place the keys randomly. So, the collector module that we design should be implemented with the facility to specify this initial layout. In other words, we should be able to specify the key width and key sizes where the layout is randomly generated.

Now, the user state predictor implements the "energy" computation. So, here you should note that we are not implementing the full FD model as we did in the previous case, but only a part of it which computes only the average movement time. So that part of the FD model is implemented in the user state predictor module.

(Refer Slide Time: 44:10)



Implementing the Framework

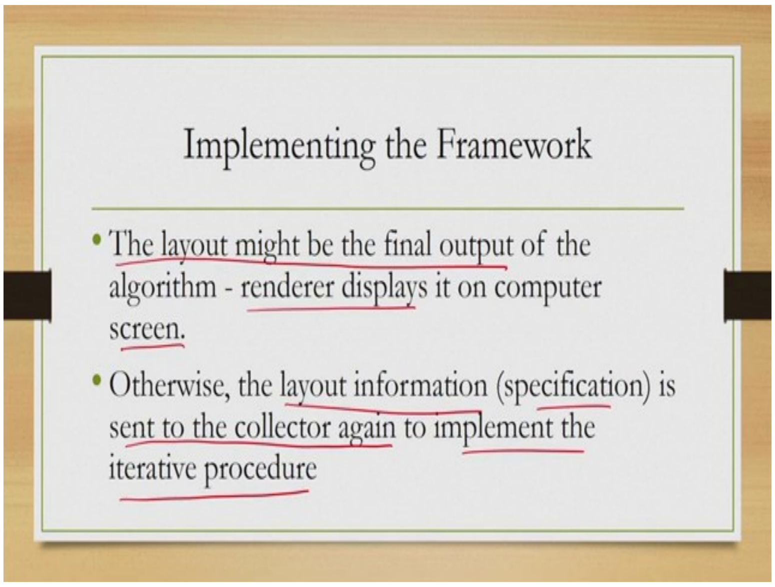
- The computed "energy" is sent to the *interface state predictor* module
- The module chooses between the previous and the new layout according to the Step 3
- From the chosen layout, it generates (specification of) a new layout on the basis of Step 1 and sends to **renderer**

Now, the computed "energy" is sent to the interface state predictor module. The interface state predictor then chooses between the previous state and the new state according to step 3 we discussed earlier that is calculation of probability and based on the threshold. So, here the implementation of the interface state predictor module is not trivial as in the previous case, here we are implementing step 3 of algorithm in the interface state predictor module.

In that step if we recollect what we did is, computed a probability and compared that value with the threshold to determine way that to keep the previous state or to discard it and replace it with a new state. Along with that we also compared the energy difference between the 2 states solve this comparisons computations are implemented as part of interface state predictor module.

Now, this module outputs the chosen layout between the previous and the newly computed layouts and it generates a specification of the new layout and sends the specification to renderer. So, we should be able to implement the interface state predictor module. So, that it can generate a layout specification indicating the key positions, distances, width and so on whatever is required.

(Refer Slide Time: 45:59)

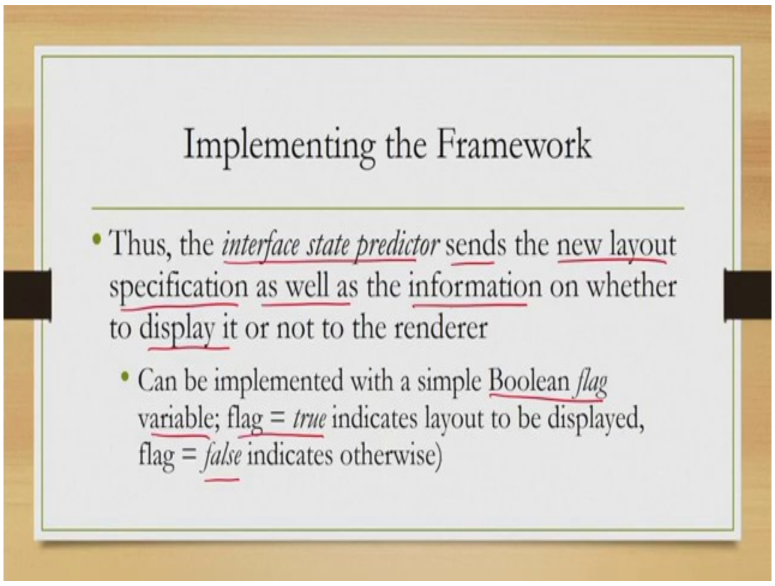


Implementing the Framework

- The layout might be the final output of the algorithm - renderer displays it on computer screen.
- Otherwise, the layout information (specification) is sent to the collector again to implement the iterative procedure

Now, this layout may be the final output so, renderer displays it on the computer screen. Otherwise, the layout information specification is sent to the collector again to implement the iterative procedure. So, the renderer needs to decide whether to display the layout as final layout or should not display and let the collector take over again to implement the iterative procedure.

(Refer Slide Time: 46:30)



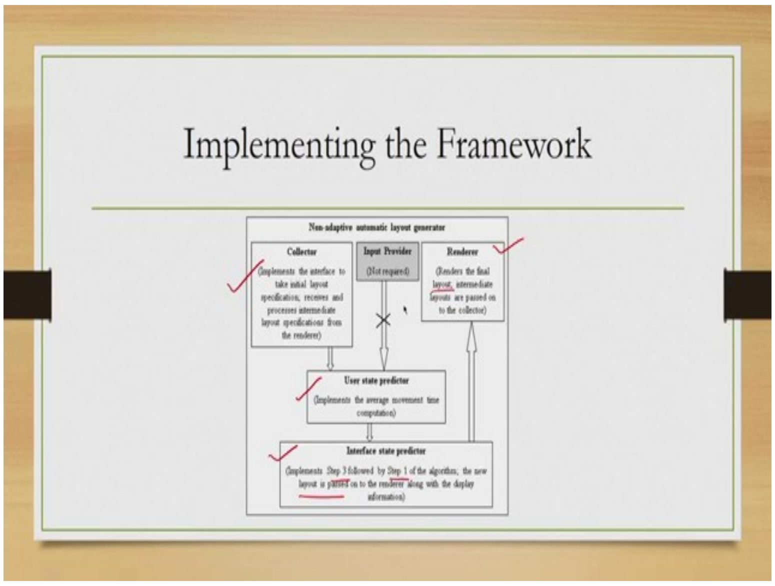
Implementing the Framework

- Thus, the *interface state predictor* sends the new layout specification as well as the information on whether to display it or not to the renderer
  - Can be implemented with a simple Boolean *flag* variable; flag = *true* indicates layout to be displayed, flag = *false* indicates otherwise)

In order to do that what we can do is that, we can use one flag actually the interface state predictor needs to send the new layout specification as well as the information on whether to display it or not. Now, to do that the predictor module can use a simple Boolean flag, true means layout to be displayed and false means renderer should not display the layout and collector should take over.

So, the renderer should check the flag and based on the flag value decides on what to do. So, that can be a simple implementation of the automated design approach for non adaptive virtual keyboard layout.

(Refer Slide Time: 47:22)



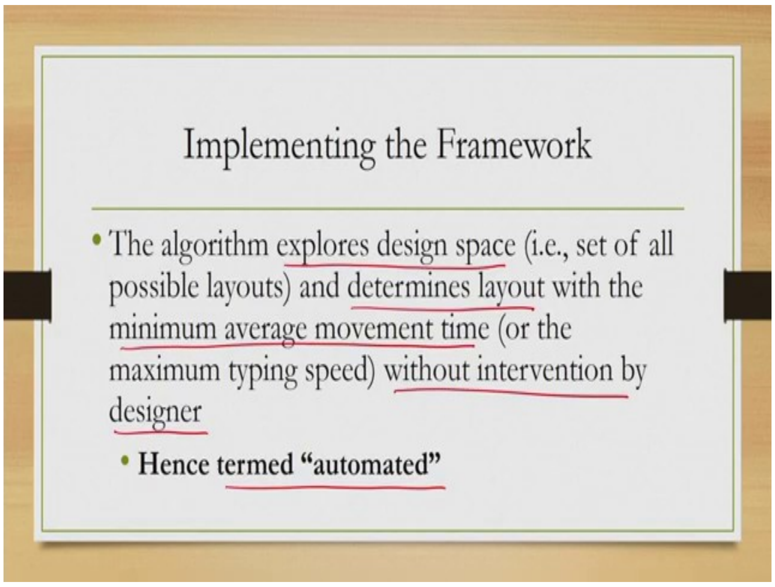
Implementing the Framework

Non-adaptive automatic layout generator
Collector
(Implements the interface to take initial layout specification; receives and processes intermediate layout specifications from the renderer)
Input Provider
(Not required)
Renderer
(Renders the final layout; intermediate layouts are passed on to the collector)
User state predictor
(Implements the average movement time computation)
Interface state predictor
(Implements Step 3 followed by Step 1 of the algorithm; the new layout is passed on to the renderer along with the display information)

So, the idea is illustrated here, the collector implements interface here to take initial layout specification the collector also receives and processes intermediate layout specifications from the renderer. So, both the tasks the collector performs and it should be designed implemented accordingly.

User state predictor implements the average movement time computation part of the Fitts diagraph model here. The interface state predictor implements the step 3 followed by step 1 of the algorithm, the new layout is passed on to the renderer along with the display information. The renderer intern renders the final layout, but intermediate layouts it does not render in that case it passes on the information to the collector so, that is the job of the renderer. So, this is how we can implement the framework to design virtual keyboard layouts using an automated approach.

(Refer Slide Time: 48:25)



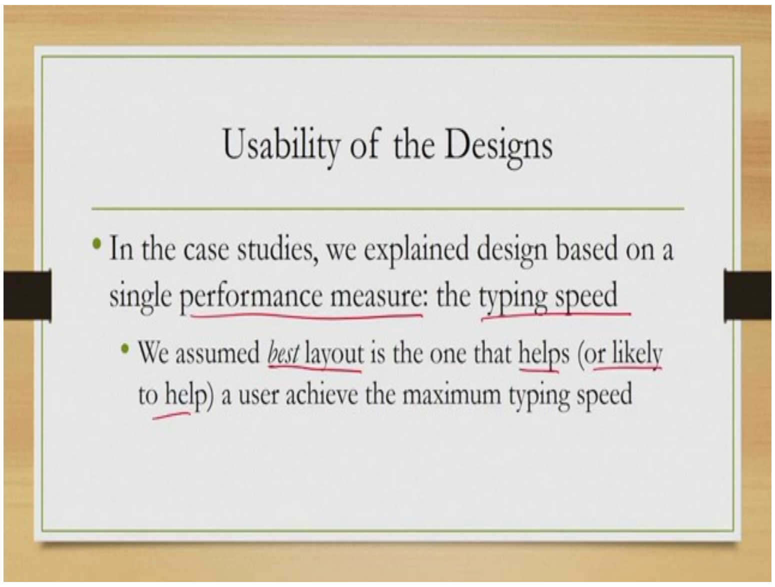
Implementing the Framework

- The algorithm explores design space (i.e., set of all possible layouts) and determines layout with the minimum average movement time (or the maximum typing speed) without intervention by designer
  - **Hence termed "automated"**

So, what the algorithm does, it is initially explores the whole design space that is the all possible designs all layouts with the specified number of keys and it determines the layout with the minimum average movement time which it can do without the intervention by the designer. So, there is no intervention by the designer hence, it is called an "automated" approach. So, the implementation of the framework takes care of this automation part by implementing the iterative process. So, with this we are expected to be able to implement the framework so that we can adopt a specific design approach.

Now, the case studies that we have covered so far are used to illustrate the idea of implementation of the framework to adopt specific design approach, out of the four approaches that we have learned earlier.

(Refer Slide Time: 49:47)

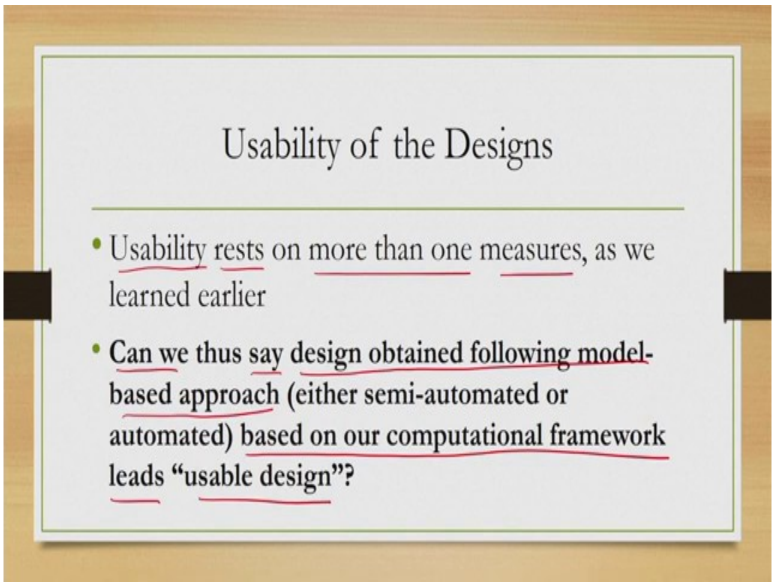


Usability of the Designs

- In the case studies, we explained design based on a single performance measure: the typing speed
  - We assumed *best* layout is the one that helps (or likely to help) a user achieve the maximum typing speed

Now, the point is in this case studies or earlier also we have seen that we are relying on performance measures, in the particular case study of virtual keyboard design we relied on only a single performance measure the typing speed. And we assumed that the best layout is the one that helps or likely to help the user achieve the maximum typing speed.

(Refer Slide Time: 50:13)



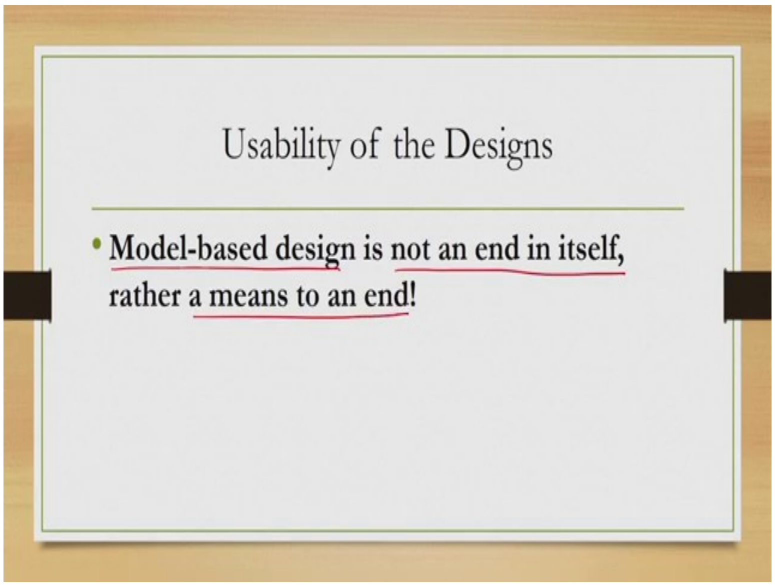
Usability of the Designs

- Usability rests on more than one measures, as we learned earlier
- **Can we thus say design obtained following model-based approach (either semi-automated or automated) based on our computational framework leads "usable design"?**

Now, if we can recollect from our earlier discussion on usability it is actually the concept that rests on more than one measures. So, the natural question to be asked at this stage is, can we say that the design that we have obtained following a model based approach which relies on computing some performance measure based on our computational framework leads to "usable design"?

So, we are using our framework to implement some design approach, but the core concept in the framework is computation of the user behavior, which we represent by implementing some computational user models where the output is some performance measures. Now, these performance measures can be indirectly related to usability as we have discussed earlier. So, the question is, can we still call our final outcome of the frame work that is the design as usable directly?

(Refer Slide Time: 51:41)

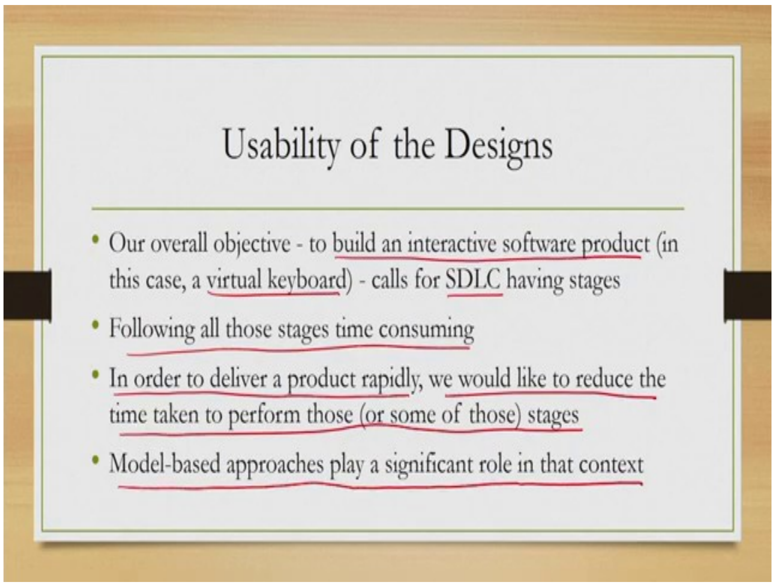


Usability of the Designs

- Model-based design is not an end in itself, rather a means to an end!

We should note that this model based design is not an end in itself, but a means to an end. So, let us elaborate, what do you mean by this.

(Refer Slide Time: 51:54)



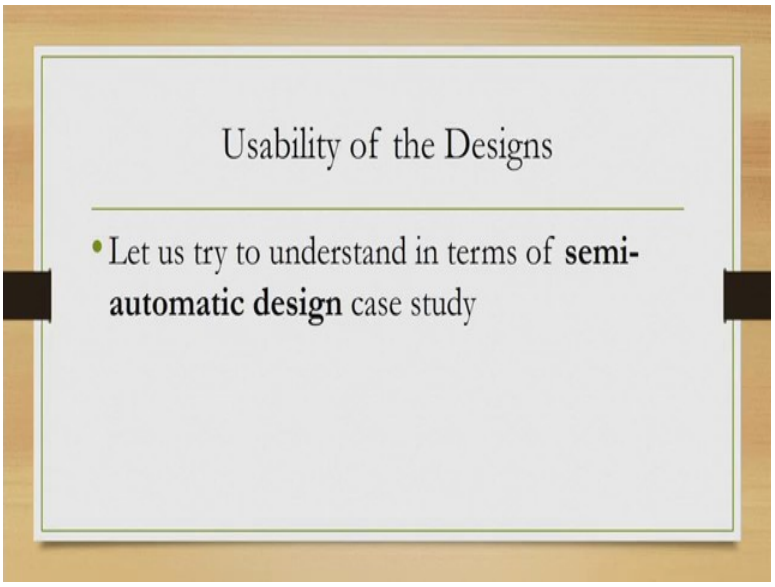
Usability of the Designs

- Our overall objective - to build an interactive software product (in this case, a virtual keyboard) - calls for SDLC having stages
- Following all those stages time consuming
- In order to deliver a product rapidly, we would like to reduce the time taken to perform those (or some of those) stages
- Model-based approaches play a significant role in that context

So, what is our overall objective, we want to build an interactive software product. So, in this case, a virtual keyboard which requires us to follow the software development life cycle stages. Now, there are several stages if you may recall and following all those stages are time consuming definitely because each stage requires lots of effort. Now, in order to deliver a product rapidly, which is a very practical requirement.

What we want, we want to reduce the time taken to perform those or some of the stages that is our wish, desire. There the model based approach can help, model based approaches can play a significant role in that context of reducing the time taken to perform software development life cycle stages.

(Refer Slide Time: 52:54)

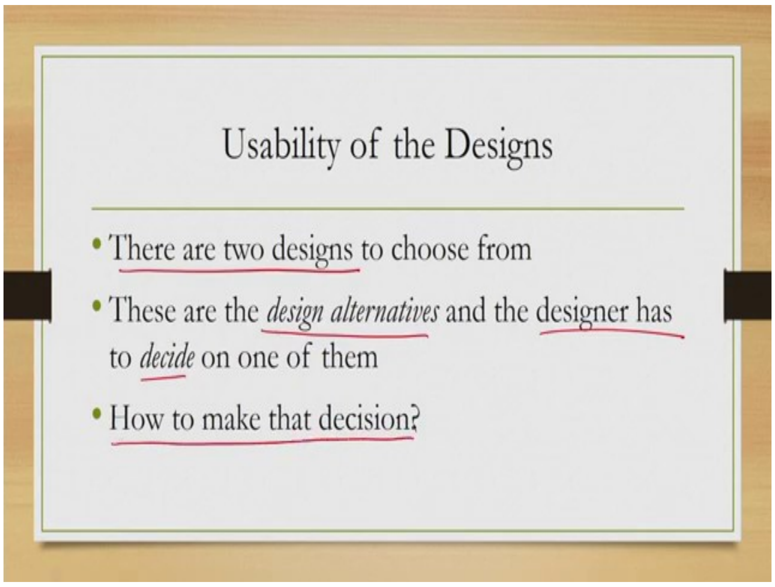


Usability of the Designs

- Let us try to understand in terms of **semi-automatic design** case study

So, let us try to understand that in terms of the semi automated design of virtual keyboards which we have discussed earlier.

(Refer Slide Time: 53:04)

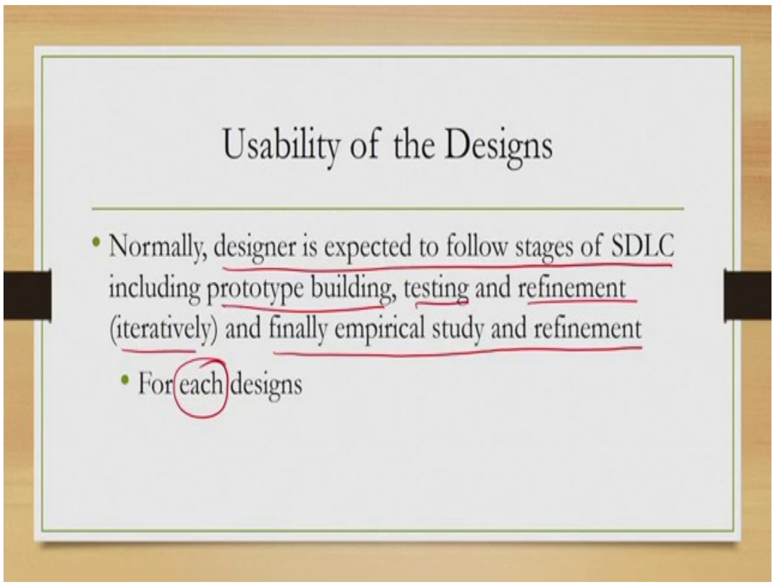


Usability of the Designs

- There are two designs to choose from
- These are the *design alternatives* and the designer has to *decide* on one of them
- How to make that decision?

So, recollect that there were two designs we mentioned, one is alphabetic and the other one is frequency arranged. Now, these are the design alternatives and the designer has to decide on one of them. Now, how the designer can make that decision? That is a very crucial question.

(Refer Slide Time: 53:25)

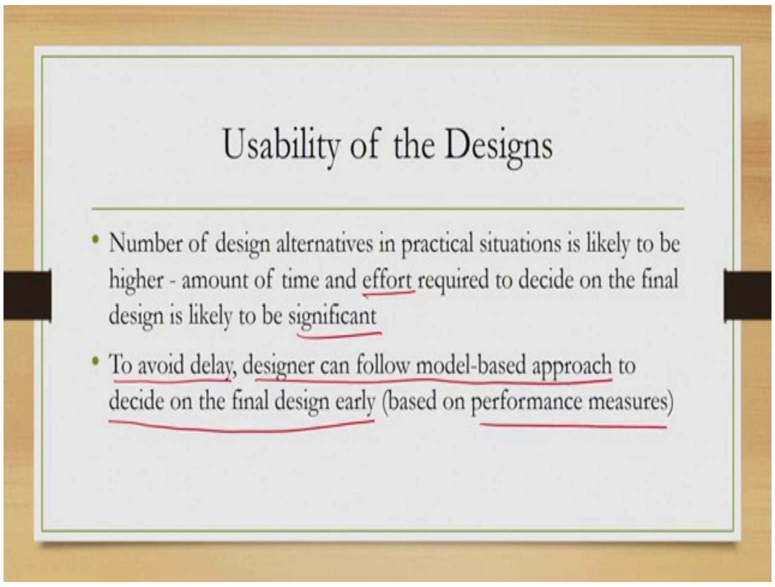


Usability of the Designs

- Normally, designer is expected to follow stages of SDLC including prototype building, testing and refinement (iteratively) and finally empirical study and refinement
  - For each designs

So, if we follow traditional approach what will happen, the designer is expected to follow the stages of the development life cycle which includes prototype building, testing and refinement of the design idea iteratively and finally, empirical study and refinement final refinement for the final design. So, this the designer has to do for each of the alternative designs. So, in our case we have considered only two but in practice there may be more number of alternative designs and the designer is expected ideally to follow the stages of the development lifecycle to find out the quality of the designs and then compare.

(Refer Slide Time: 54:16)

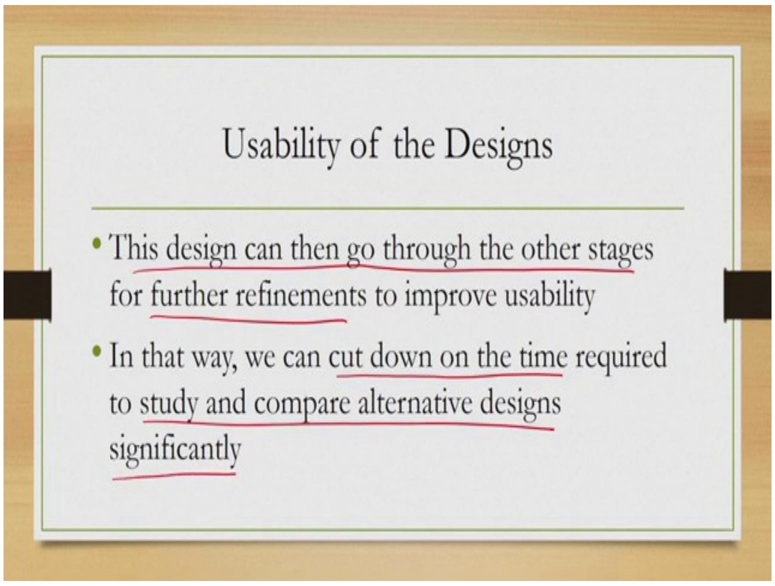


Usability of the Designs

- Number of design alternatives in practical situations is likely to be higher - amount of time and effort required to decide on the final design is likely to be significant
- To avoid delay, designer can follow model-based approach to decide on the final design early (based on performance measures)

Clearly that takes time and the time is likely to be significant, it also involves effort it also involves costs. So, we are for the time being ignoring those parts, but all these effect the overall outcome of the design process. Now, what the designer can do, to avoid the delay the designer can follow the model based approach to decide on the final design at an early stage of the design life cycle and this decision will be based on the performance measures that can be obtained by implementing the proposed framework.

(Refer Slide Time: 54:56)

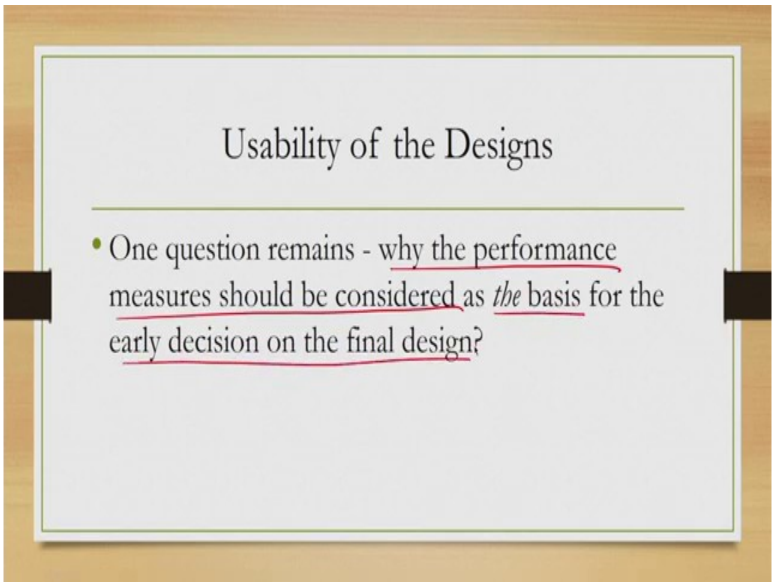


Usability of the Designs

- This design can then go through the other stages for further refinements to improve usability
- In that way, we can cut down on the time required to study and compare alternative designs significantly

And then the design can go through the other stages for further refinements. So, if we can identify the problems early if we can identify a design early without going through all the stages then we can focus on that design only and perform the stages to refine that design to have better usability. And this is likely to cut down the time required for the overall design by cutting down the time required to study and compare alternative designs in a significant way.

(Refer Slide Time: 55:34)

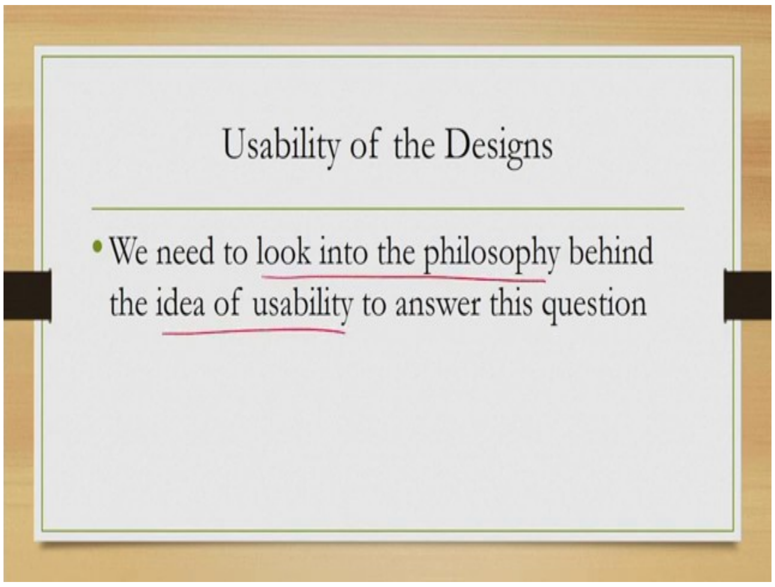


Usability of the Designs

- One question remains - why the performance measures should be considered as *the* basis for the early decision on the final design?

But that still does not answer basic question, why the performance measures should be considered as the basis for the early decision on the final design? So, we mentioned that using the implementation of the computational framework the designer can compute performance of a design and compute that for many designs then compare and refine it by following standard approach. However, the first stage that is comparison relies on one or few performance measures which are indirect measures of usability. So, why to rely on these measures to determine usability?

(Refer Slide Time: 56:14)

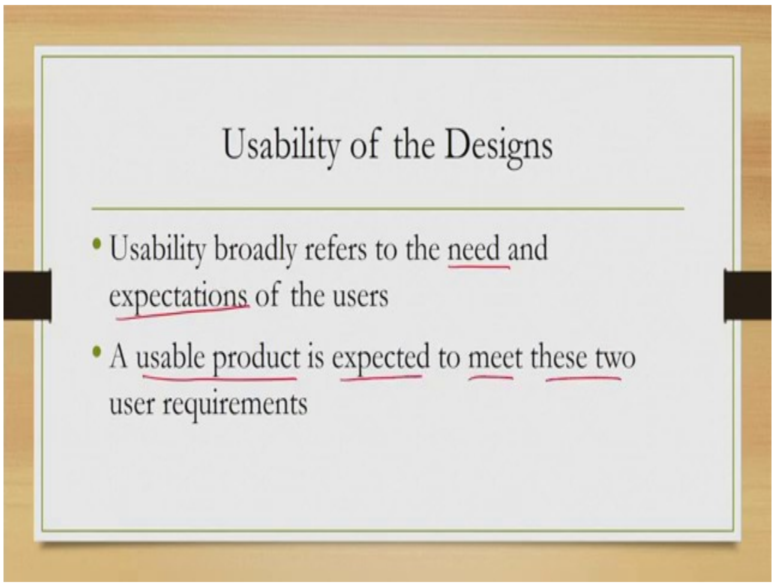


Usability of the Designs

- We need to look into the philosophy behind the idea of usability to answer this question

To answer this question we need a deeper thinking, we need to look into the philosophy behind the idea of usability and what is that.

(Refer Slide Time: 56:23)

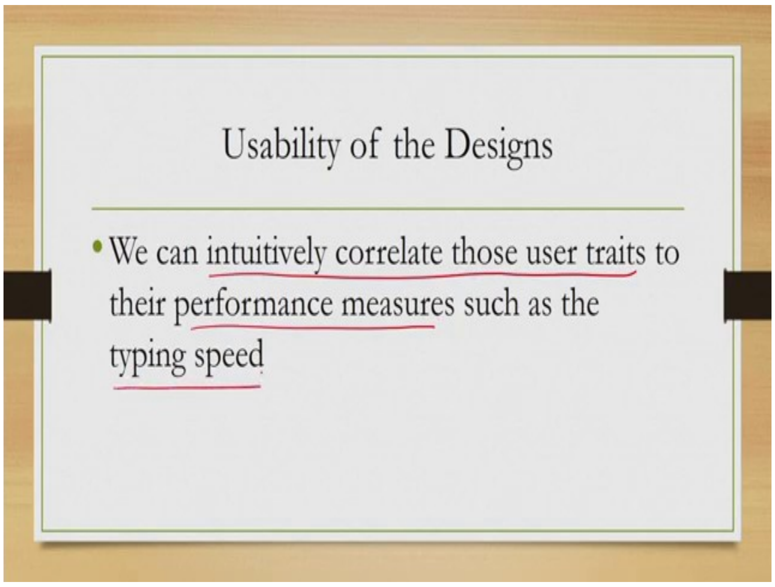


Usability of the Designs

- Usability broadly refers to the need and expectations of the users
- A usable product is expected to meet these two user requirements

So, let us forget the formal definition. So, what essential usability means in a very basic term usability refers to the need and expectations of the user. A usable product is expected to meet these two user requirements. When we talk of usability we are actually referring to the ability of a system or a product to meet the needs and expectations that are there for user before he or she goes to use the product or during the usage of the product.

(Refer Slide Time: 57:03)

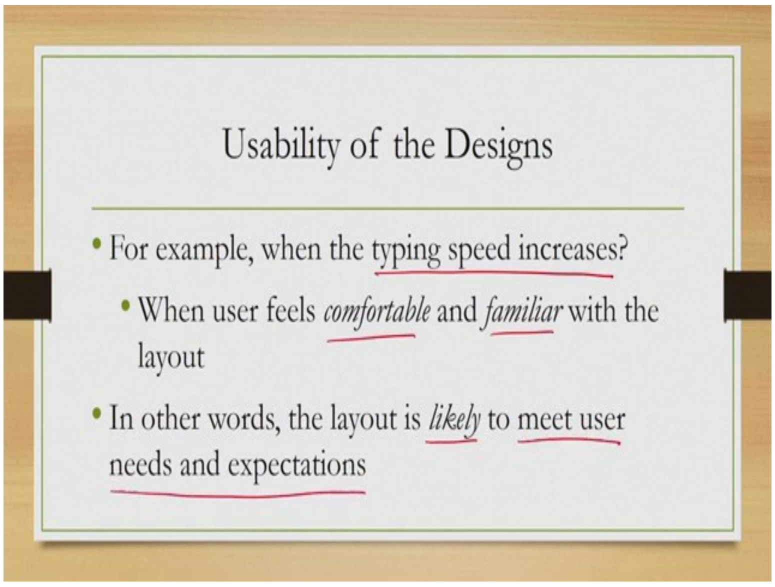


Usability of the Designs

- We can intuitively correlate those user traits to their performance measures such as the typing speed

Now, we can intuitively correlate those user traits namely the need and expectations to their performance measures such as typing speed for keyboard designs, why.

(Refer Slide Time: 57:21)

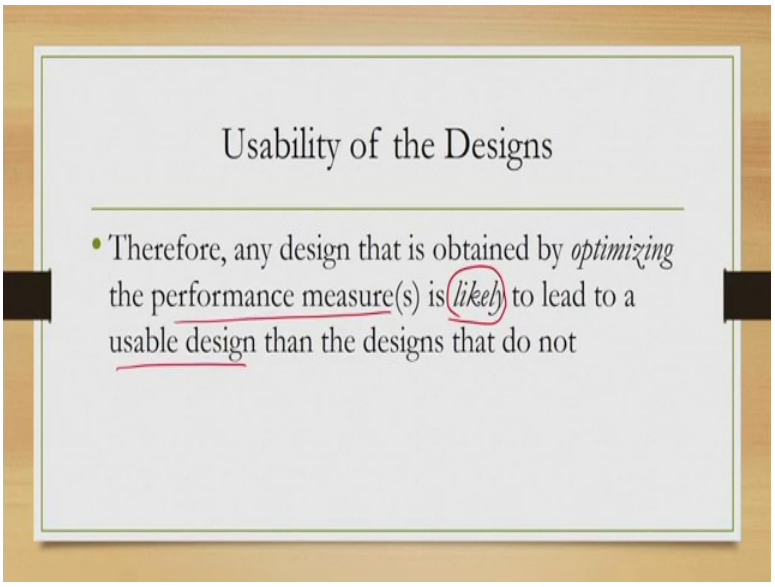


Usability of the Designs

- For example, when the typing speed increases?
  - When user feels *comfortable* and *familiar* with the layout
- In other words, the layout is *likely* to meet user needs and expectations

Let us take an example. So, when the typing speed increases intuitively we can say that when the user feels comfortable and gets familiar with the layout. Now, this comfortability with the layout of familiarity with the layout are related to the needs and expectations of the. So, if we see that there is an increase in typing speed then it is likely that the layout is able to meet user’s needs and expectations.

(Refer Slide Time: 57:57)

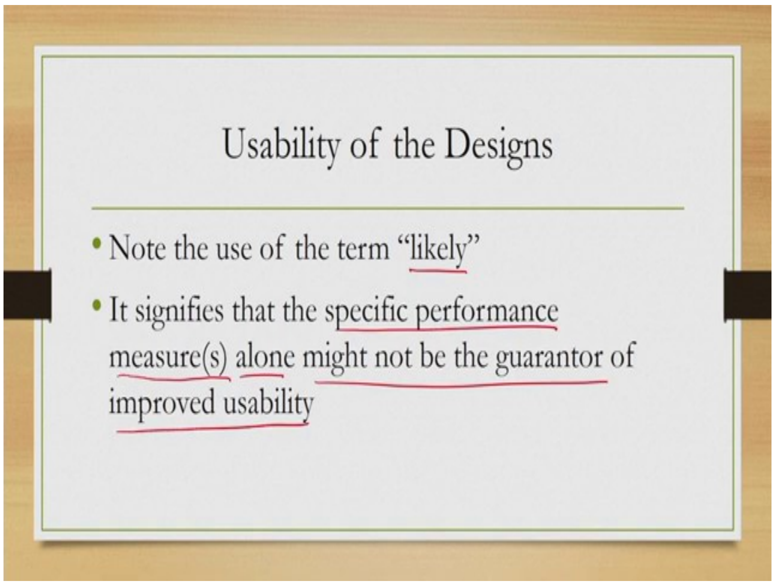


Usability of the Designs

- Therefore, any design that is obtained by *optimizing* the performance measure(s) is *likely* to lead to a usable design than the designs that do not

Now, here we are trying to optimize the performance measures. So, the assumption is that the design that is obtained by optimizing the performance measures is likely to lead to a usable design then the designs that do not. So, if you have a design where the performance measure is not optimized then it is likely to be less usable than the design where the performance measure is optimized.

(Refer Slide Time: 58:23)

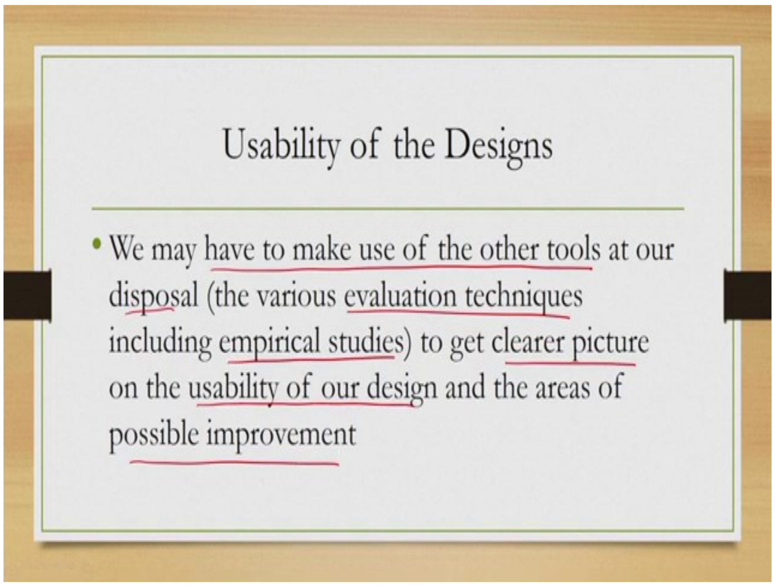


Usability of the Designs

- Note the use of the term "likely"
- It signifies that the specific performance measure(s) alone might not be the guarantor of improved usability

Note the use of the term “likely”. So, this term signifies that specific performance measures alone may not be the guarantor of improved usability. So, here we are indicating some possibility rather than certainty.

(Refer Slide Time: 58:44)

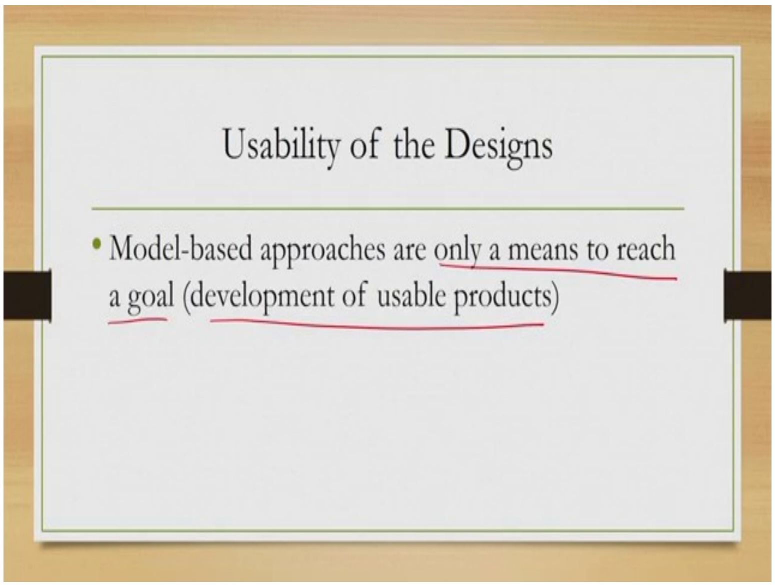


Usability of the Designs

- We may have to make use of the other tools at our disposal (the various evaluation techniques including empirical studies) to get clearer picture on the usability of our design and the areas of possible improvement

And in order to come to a certain conclusion we cannot rely only on the model based approach. What we need to do, we have to make use of other tools at our disposal those are various evaluation techniques including empirical studies to get a clearer picture on the usability of our design and the areas of possible improvement. So, in a later lecture we will talk about those evaluation techniques in details.

(Refer Slide Time: 59:15)



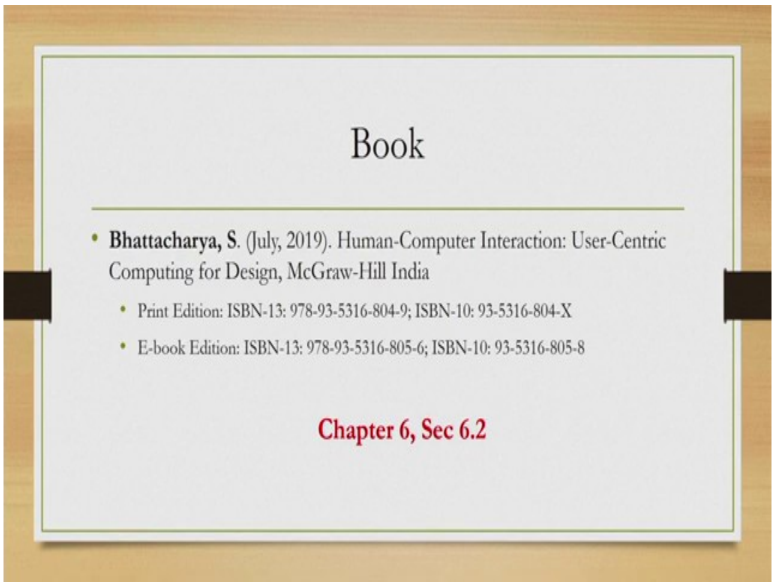
Usability of the Designs

- Model-based approaches are only a means to reach a goal (development of usable products)

But for the time being what we should note is that model based approaches are only a means to reach a goal what is our goal to develop usable product and with model based approach we can reach to that goal with the help of other techniques also. Only with model based approach we cannot reach to that goal because we cannot say with certainty whether the end product will be usable or not, for that we need to use other evaluation techniques in conjunction with model based approaches.

So, we have got some idea of what we mean by the model based approach, how to implement it in the context of interactive system design and why this implementation of model based approach for interactive system design is going to be beneficial as well as going to be useful in meeting our overall objective of coming up with a usable product.

(Refer Slide Time: 60:18)



Book

- **Bhattacharya, S.** (July, 2019). Human-Computer Interaction: User-Centric Computing for Design, McGraw-Hill India
  - Print Edition: ISBN-13: 978-93-5316-804-9; ISBN-10: 93-5316-804-X
  - E-book Edition: ISBN-13: 978-93-5316-805-6; ISBN-10: 93-5316-805-8

Chapter 6, Sec 6.2

The material that we have covered today is taken from this book the same book that we are following for all the lectures, you can refer to chapter 6 section 6.2 to get more details all the references for all the models and algorithms that we have discussed here.

So, in the next lecture we will start on another important topic in user centric computing for human computer interaction namely empirical studies.

Till then thank you and goodbye.