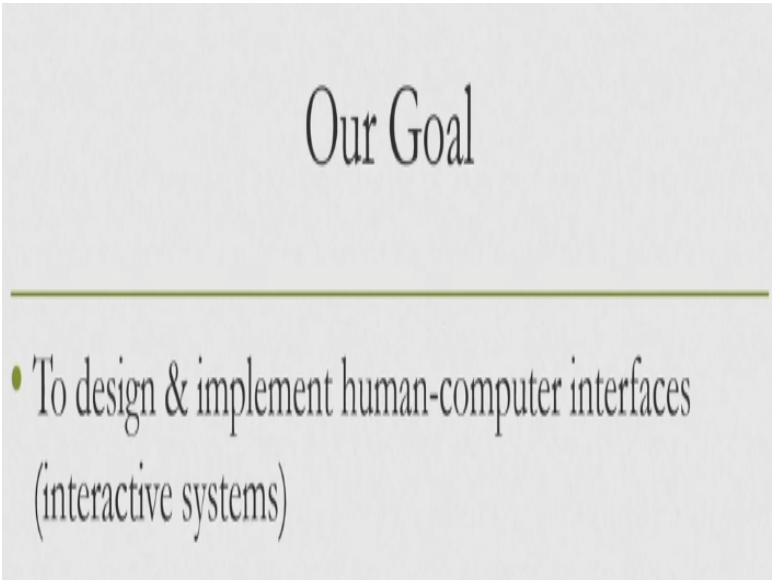
Design and Implementation of Human-Computer Interfaces
Dr. Samit Bhattacharya
Department of Computer Science and Engineering
Indian Institute of Technology – Guwahati

Lecture – 44
Concluding Remarks

Hello and welcome to the NPTEL MOOCS course on design and implementation of human-computer interfaces. We are going to start our last lecture concluding lecture that is lecture number 38. In the previous 37 lectures as well as some additional addon lectures on case studies, we have covered several key concepts and materials related to the design and development process of interactive systems. So, what was our goal in this course?

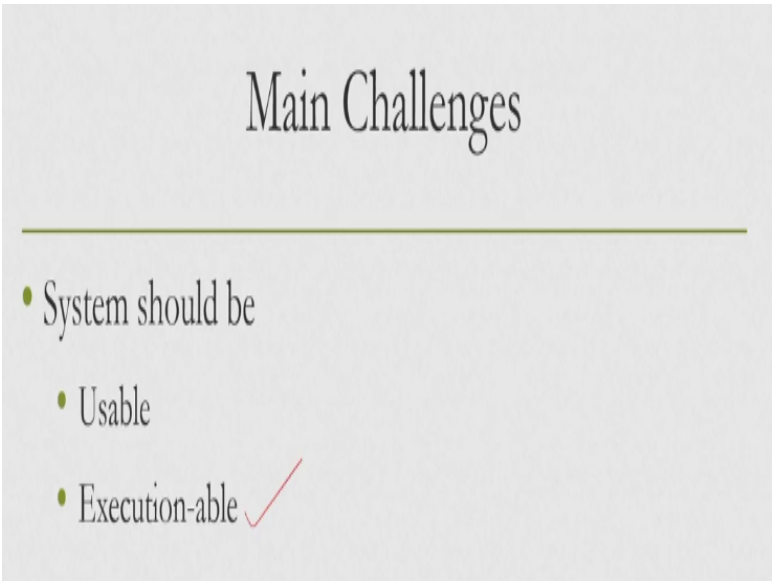
(Refer Slide Time: 01:29)



At the beginning, we mentioned that what we want to learn in this course is how to design and implement human-computer interfaces. Now, human-computer interfaces are examples of systems that are called interactive systems. So, in order to develop and implement, in order to design and implement human-computer interfaces what we need is to know how to design and implement interactive systems.

Here by the term system what we are referring to is a software that is to be run on some hardware, but hardware design is not our concern here, our concern is to develop software. So, when we talk of developing a software. What is the difference between design and implementation of any software and design and implementation of interactive software? When we are talking of interactive software, we have primarily two concepts.

(Refer Slide Tim: 02:36)



In general, when we are talking of developing any software, the idea is to make it efficient from the point of view of execution. So, whether the software is executable and whether the software is executable in an efficient manner that is the primary concern that is the concern that we also have when we are talking about interactive system development and implementation.

Along with that, we have another crucial concern that is the concern of usability. So, whether our software is going to be usable. Remember that earlier we talked about the definition of usability that is the software should be effective, efficient and satisfactory to specified group of users in specified context of use to achieve specified goals. This is the ISO definition of usability. So, when we are building a software, we need to ensure that the software is effective.

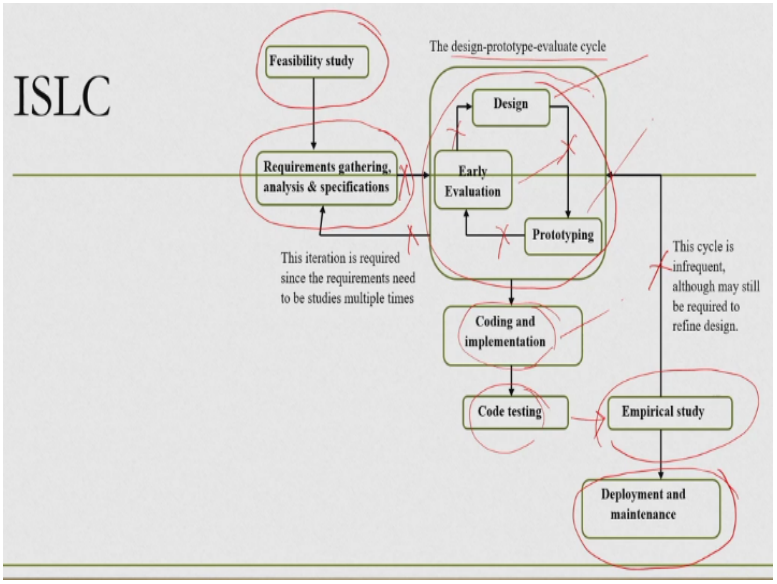
Efficient and satisfactory to the target group of users to achieve a specified set of goals that the users may have in a specified work context. So, both are very important concerns when we are talking of design and implementation of interactive systems. Whereas, for other systems any software which is not interactive, the primary goal is to design a system that is executable in an efficient way.

In order to ensure that we can achieve these objectives of ensuring these two things usability and execute stability, we need to follow some systematic development approach random ad hoc on the fly approach may not work, mostly it will not work if we are talking of complex

real world systems. So, some systematic development approach needs to be followed which brings us to the concept of interactive system development lifecycle.

In general that is one type of software development lifecycle, which we have studied in this course. And we have specifically studied one such development lifecycle which we are calling interactive system development lifecycle. It contains several stages.

(Refer Slide Time: 04:29)



First phase is the feasibility study stage. This stage we have not covered much, we just mentioned it in the passing it is not related to our main purpose of this course. We started our discussion with the next stage that is requirement gathering analysis and specification stage. This is followed by the cycle design, prototype, evaluate cycle as shown here So, in this cycle, we go for design of the interface, user interface as well as the user interaction with the system.

We prototype that design for quick evaluation. Based on the evaluation results, we may need to refine the design, again we prototype that refined design, go for evaluation, and this cycle continues till we arrive at a stable design. This is followed by a system design, where we primarily aim to come up with a modular system design and we can use functional or object-oriented approaches to design and express the system.

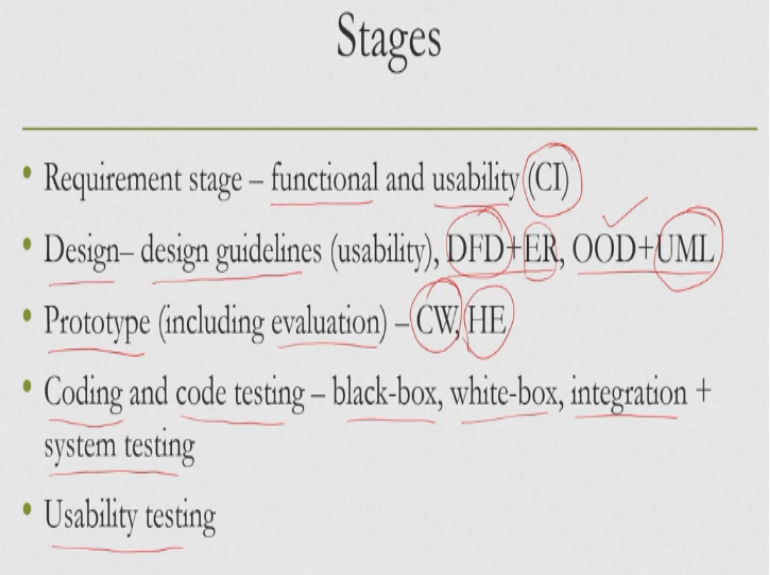
This is followed by the next stage that is coding an implementation. We have discussed this stage in details followed by the testing of the code, again we have covered it in significantly details. This is followed by the empirical study, covered in details. Now, there are several

cycles or iterations as you can see. There is an iteration between this design, prototype, evaluate cycle to requirement gathering cycle as shown with these two arrows marked in cross, this should not be frequent.

The meaning of this cycle is that if in the design of interface stage, we find some issues which are not being getting resolved then we may need to refine the requirements. We may go for newer requirement gathering, so this cycle may happen, but it should not be frequent. Then we have this cycle design, prototype, evaluate cycle marked with these crosses, this may be frequently done.

Its purpose is to execute frequently so that we can arrive at a stable interface design. There is this cycle also between empirical study to the design stage and this cycle also should not be frequent. At the end, we have this deployment and maintenance stage, this stage also you mentioned in brief. So, what we have covered in those stages?

(Refer Slide Time: 07:22)



In the requirements stage, we collect functional and usability requirements. Usability requirement is collected through several methods, application of either of several methods. One method we discuss that is contextual inquiry. In the design phase, for interface design we may follow some design guidelines to ensure usability of the interface. For system design, we may follow functional approach with the DFD as the language to express the design.

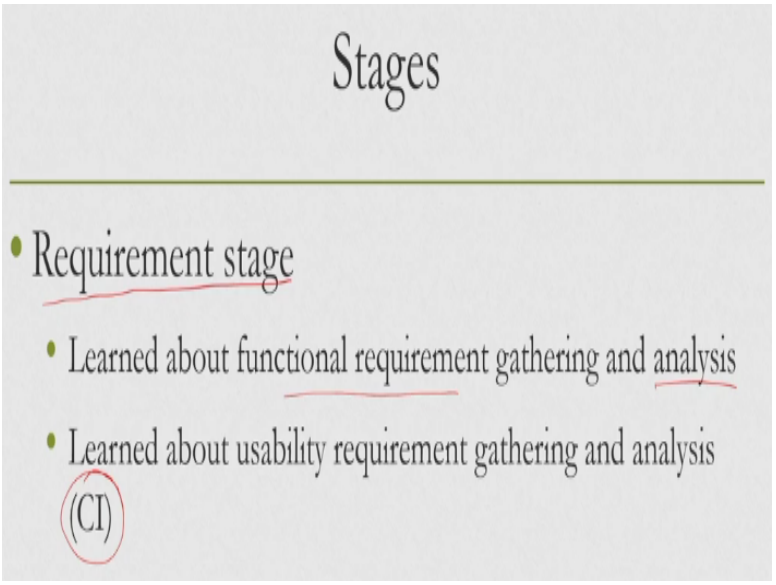
Or we can follow the object-oriented design approach with UML as the language to express the design. So, UML is the language to express object-oriented design whereas DFD is the

language to express functional design. Additionally, we can use the ER or entity relationship as part of the design to express the structure of databases in the design in functional approach. In the prototype stage, we have covered several ways to create prototypes.

And also we have seen how to evaluate those prototypes quickly using either of cognitive walkthrough or heuristic evaluation methods. We have learned in details these methods. In the coding stage, we have seen several good coding practices and we have talked about what to do and what not to do during coding. In code testing, we have covered review-based testing, black box testing, white box testing.

We have also seen integration of different units or modules into a full system and testing the full system for functionality as well as performance. Finally, we have covered usability testing in details.

(Refer Slide Time: 09:09)



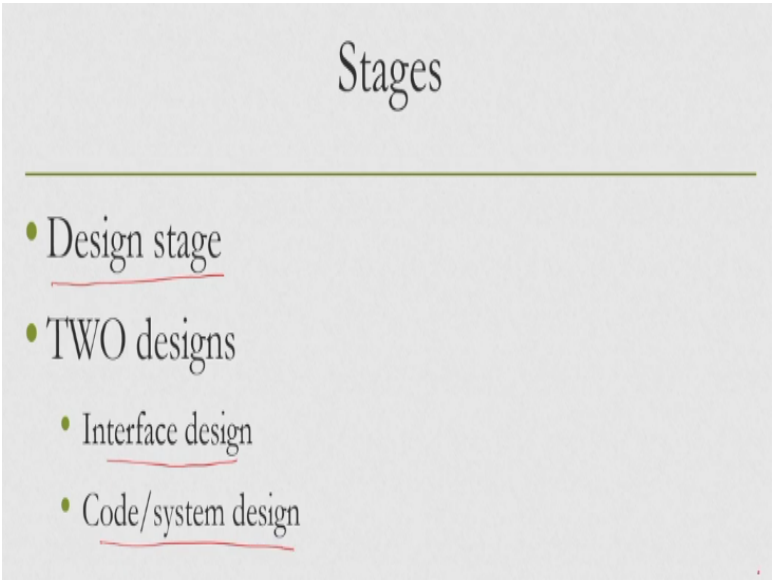
So, let us quickly go through the concepts that we have learned in each of these stages. In the requirements stage what we have covered? We have learned about how to gather functional requirements and how to analyse those requirements to come up with the specifications. Also, we have learned about the idea of non-functional requirements, in particular usability requirements, because that is the very crucial concept with respect to the scope of this course.

So, you have learned about usability and we have learned how to gather usability requirements. There we talked about the contextual inquiry method. In the contextual inquiry

method if you recollect, there are 5 stages starting with planning, then approaching the management that is called initiate, then execution that means actually collecting data.

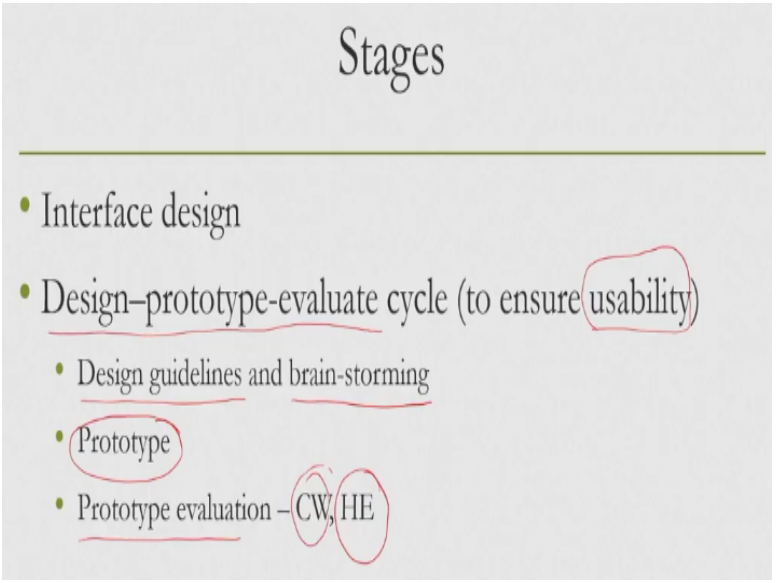
Then close that means sending thank you notes to the participants as well as organizations and finally reflect or analysis of the collected data. So, we have seen through several case studies how the data can be collected, recorded and analysed.

(Refer Slide Time: 10:25)



Next, we covered the design stage. So, there we actually talked of two types of design, one is the interface design, other one is the code or system design. So, in interface design what we have covered?

(Refer Slide Time: 10:38)



The design, prototype, evaluate cycle that we have just mentioned in the previous slide is part of the interface design stage. Now, we carry out this cycle to ensure usability of the design. Usability is our primary concern. So, we ensure usability by performing this design, prototype, evaluate cycle. Now, in the design stage design of the interface, we covered some design guidelines.

So, when we want to design some interface either we can rely on our intuition or experience or design guidelines or a combination of all these three. So, we covered several guidelines, particularly two sets of guidelines, Shneiderman's eight golden rules and Norman's seven principles, these are generic guidelines. We have also seen there can be specific guidelines, specific to products.

With guidelines we need to brainstorm to come up with interface design, this is followed by building of prototype. So, we have seen that that prototypes can be of different types. We can have low fidelity prototype which does not require any use of sophisticated technology including computers, then we can have medium fidelity and high fidelity prototypes. We have also seen the concepts of vertical prototypes and horizontal prototypes.

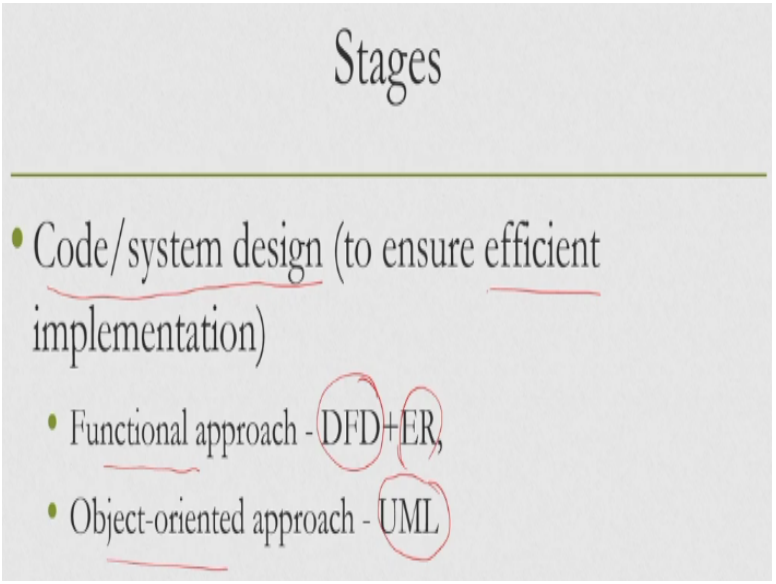
As well as how to use the prototypes incremental way, evolutionary way, throw away prototypes; all these things we have covered. We have covered in details how to evaluate the prototype quickly with experts who are supposed to represent the behaviour of the target user group. Why we are going for expert evaluation? Because we want quick evaluation as we have noted later in empirical study, if we need to get it evaluated with end users it requires significant time and effort.

But in the early stage, we want to find out problems quickly so that we can refine and perform the cycle several times before we stabilize the design, so they are we rely on experts for quick evaluation with feedback. And for such an evaluation we covered in details these methods cognitive walkthrough and heuristic evaluation. In cognitive walkthrough, we start with usage scenario, typically representative usage scenario, ask the evaluators to carry out those tasks in the scenario and prepare reports.

Typically, the team contains 3 to 5 members. Similar type of teams we use for heuristic evaluation, but here instead of usage scenario, we provide the evaluators with checklist and

they check the system against the checklists and prepare reports. We have seen the 10 heuristics by Nielsen as one of the checklists and we can use that for heuristic evaluation purpose. So, all these things we have discussed in details during the relevant lectures.

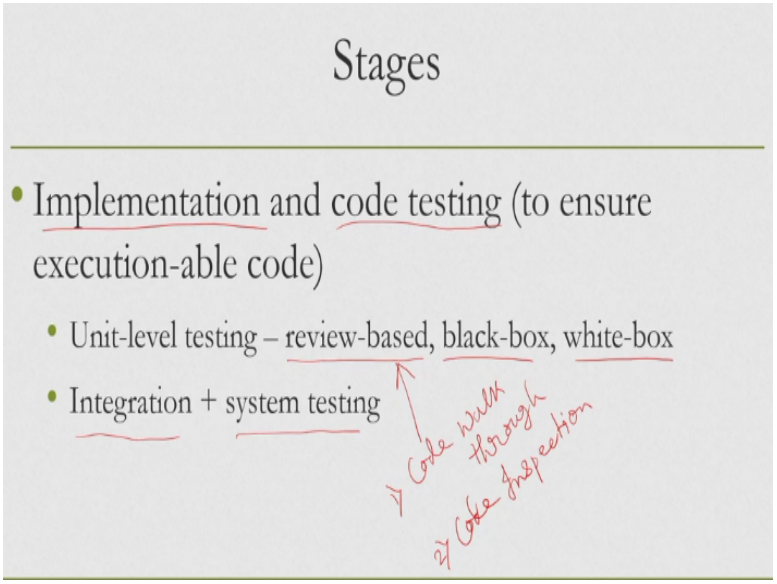
(Refer Slide Time: 13:54)



Other thing that we have learned in details is how to go for code or system design. The objective is to ensure efficient implementation of the system. So, we have seen that the primary way to do this is to think of the system as in the form of a hierarchy of modules with well-defined connections between the modules. Now, we can approach this design in either functional way or object-oriented way.

If we are following an object-oriented approach, then to express our design we can use UML, unified modelling language. If we are following a function-oriented approach, then to express the corresponding design we can use DFD in combination with your ER where DFD stands for data flow diagram, ER stands for entity relationship diagram. We can use these two together to express the design. All these things we have discussed in significant details.

(Refer Slide Time: 15:05)



Next stage that we have covered is the implementation stage. In implementation, we have learned about good coding practices, what we should do, when we are going for implementing a system, also what we should not do when we are going for implementing a system. There we have learned about some simple rules like length of a function in terms of number of instructions should not be more than 10 ideally, naming conventions of variables of functions, how to use global variables and so on and so forth.

This was followed by detailed discussion on testing of the code. So, once we are able to implement the system by writing code, next we need to test the code to see whether there are any bugs, whether there are any issues. There is a quick way of testing and there is a rigorous way of testing, we discussed both. The quick testing involves review-based testing and rigorous testing involves more systematic way of testing.

Now, when you are developing what typically happens is that first unit level developments take place that means at the module level, then the modules or units are combined together into the system and the system level testing takes place. For unit level testing, we can use review-based testing methods for quick testing to quickly know about the issues with the code so that we can refine and go for more rigorous testing.

Two set testing methods we have seen, one is code walkthrough and the second one is code inspection. Now code walkthrough; conceptually similar to the concept of cognitive walkthrough that we have seen earlier in the context of prototype evaluation. Like in cognitive walkthrough in code walkthrough, we start with a carefully selected test cases, one

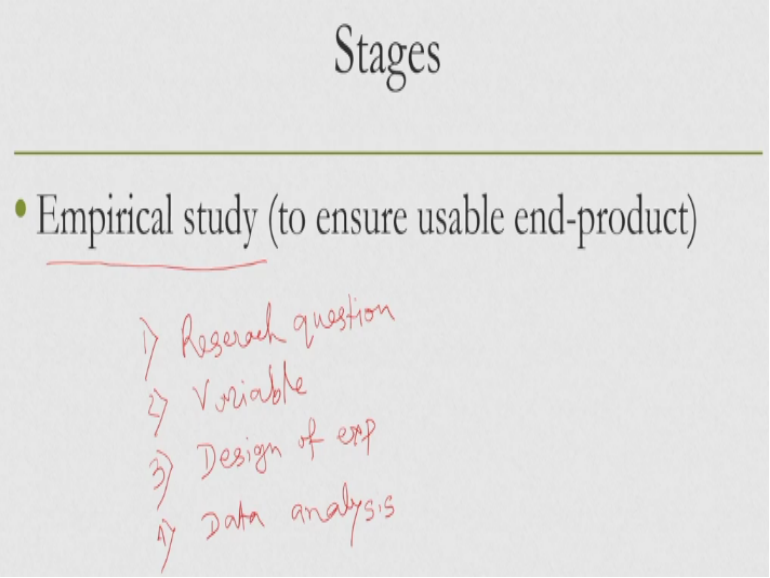
or more such test cases and manually or through visual inspection we try to execute the code for the cases and see whether there is any issue.

In code inspection we check the code for violation of standards and guidelines. This is similar in concept to the heuristic evaluation or checklist-based evaluation for prototypes. Then, we have learned also in details about several rigorous methods namely the black box testing method and the white box testing method. In black box testing, the key thing is to divide the input domain into equivalence classes and then choose the appropriate test cases, also you need to perform boundary value analysis.

In white box testing, we decide on the test cases based on either of several approaches. We primarily learned about path coverage approach and how it can be used to decide on test cases. So, those are unit level testing. Also you have learned about integration and system level testing including use of driver and stub modules for performing system level testing while all the units are not integrated.

So, we are broadly seen that two approaches bottom-up and top-down approaches. And also we have seen functional and performance testing of whole system as well as the other concepts of testing, namely alpha testing, beta testing and acceptance testing.

(Refer Slide Time: 19:32)

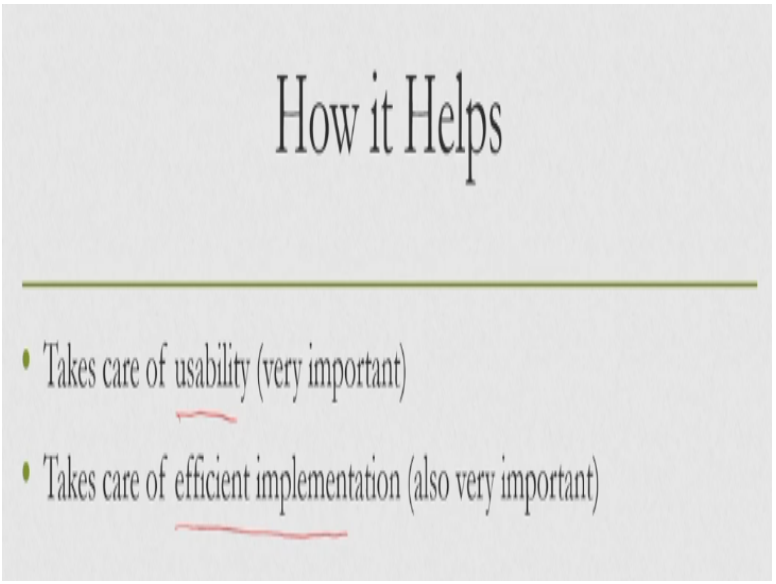


Finally, we talked about empirical study where our objective is to test the usability of the product. So, there are four stages of the study that we have learned in detail. One is research question formulation. Stage number 2 is variable identification then in stage number 3 design

of experiment is done and fourth stage is data analysis. So, these are the stages through which we carry out the empirical study.

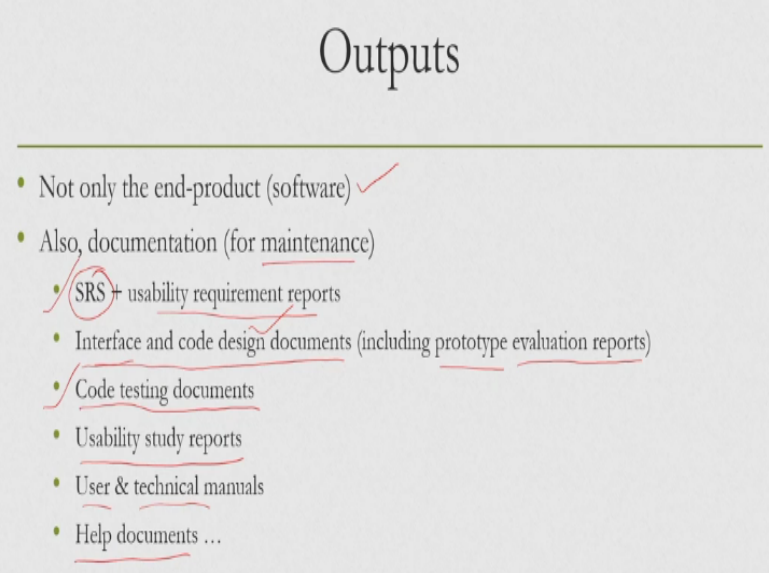
And at the end of data analysis, we conclude about the usability of the product. Here of course, data analysis is not a simple thing, in fact none of the stages can be done in a very ad hoc and simple manner, it requires careful considerations as we have seen during the lectures. So, once we follow all the stages to develop the system, what is the benefit that we get?

(Refer Slide Time: 21:11)



Now, if we follow these stages, then we can be sure that the product is usable depending on the conclusion drawn and the product is executable that means the product can be implemented in an efficient manner in terms of use of resources. So, both are important and by following this lifecycle both can be ensured if we follow them properly.

(Refer Slide Time: 21:38)



But the following of stages is not the only thing and the development of the system is not the only thing, along with that at every stage some documentation is done. Now, this documentation has to be done very carefully, very elaborately so that the overall development process is recorded and equally crucially the development process can be understood by others. So, the documents should be understandable and maintainable for future maintenance and bug fixing.

So, the software is not the only end product, along with that the documents are also there for maintenance in future and there are several documents, not a single document. After the requirements stage, we get this SRS plus usability requirement reports. SRS stands for software requirements specification. After this design phase, we get this interface and code design documents which includes documentation on prototype building.

And evaluation of prototypes, reports related to the evaluation of the prototype including the team composition, the tasks that are given, the feedback received, compilation of those feedbacks, etc., in every cycle. In case of code design, we create these design documents either in the form of DFD or UML. Then, we generate code testing documents, for each type of testing documents should be generated.

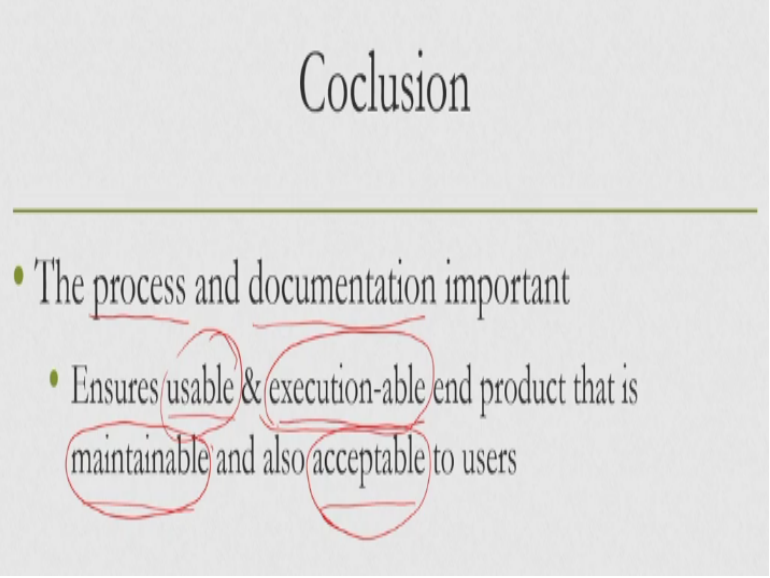
For example, if you are performing review-based testing also then documentation related to that, if you are performing black box testing documentation related to that, if you are performing white box testing documentation related to that, everything you need to generate.

Now, usability study reports that is empirical study reports has to be produced at the end of empirical study stage outlining in details everything related to the study.

Including research questions, hypotheses, experimental procedure setup, design of experiment and how the data were analysed. Along with that, we need to also create the user documents, user manuals and technical manuals. So, user manual is to help the user understand the system features and technical manual is required to further understand the code as well as update the code if required.

Help documents are also useful that is part of user manuals, you can think of it as part of user manuals. So, we have covered several case studies to see how these documents look like. We have explicitly discussed how these documents look like for SRS, four testing documents, for design documents which are primary components of these types of documents. So, along with the software, this document creation is also very important for recording the process for future maintenance purpose.

(Refer Slide Time: 25:14)



So, both are important; the execution of the process as well as creation of the documents. If we follow the process properly with required rigor and if we create the documents properly then that ensures that we end up with a usable and executable end product that is maintainable and also acceptable. So, four key words are important here; usable, executable, maintainable and acceptable.

All these four are important and if we follow these stages in a systematic manner with all the required rigor and the documentation properly, then we can ensure that all these four keywords are satisfied. However, as we have discussed in the previous lecture, sometimes following this development lifecycle may require long time and involve cost which may not be acceptable to the end users, in that case we may have to go for some compromise, we may have to go for agile development methods that is also possible.

So, with that we have come to the end of this course. I hope you have learned all the concepts. I hope the lectures were good learning experiences and I also hope that by going through these lectures, by covering this course, you are now in a better position to understand the development process of an interactive system. And also, you are now ready to develop and deploy your own system which is a human-computer interface or in other words another interactive system.

For example, if you are now trying to make a game, a video game, then you know what are the things to do and you are expected to follow those things so that your game is acceptable to those who want to play such games. You must remember that whenever you are building a product and usable product, it need not be usable to everybody. So, if you are building a game, it need not be visible to everybody, only to those who are interested in similar type of games.

So for that you need to properly identify the requirements and you need to properly go through all the stages to develop the usable end product which is also executable. We generally have this tendency of only focusing on executability of the software, whether it can run on a given system rather than focusing on usability. So, through the lectures and throughout this course, my objective is to also emphasize on the point that along with executability, a software which is interactive needs to be usable also.

So for that whatever is needed to be done you have to do. And I hope by now you are quite aware of the things that need to be done to make the product usable as well as executable and you will be able to translate that knowledge into practice. With that hope I would like to end this lecture as well as this course. Hope you have enjoyed the course. Thank you and goodbye.