Design and Implementation of Human-Computer Interfaces

Dr. Samit Bhattacharya
Department of Computer Science and Engineering
Indian Institute of Technology – Guwahati

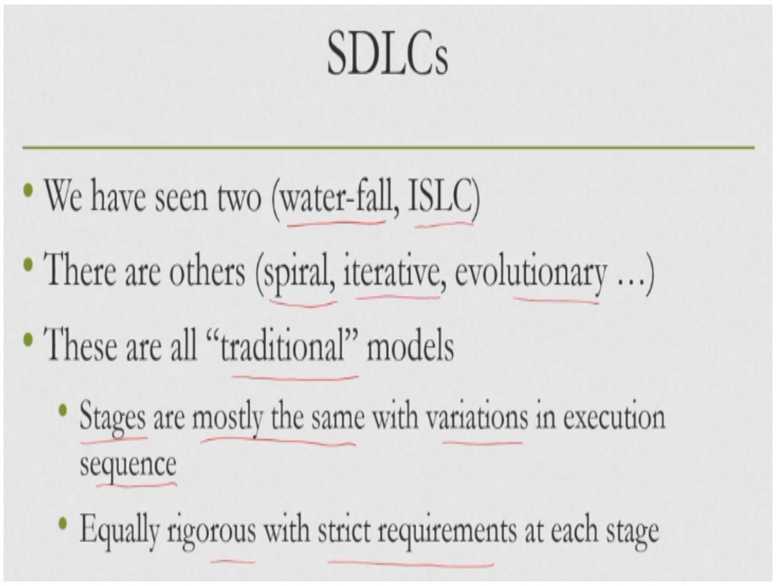
Lecture – 43
Note on Agile Development

Hello and welcome to the NPTEL MOOCS course on design and implementation of human-computer interfaces. We are going to start lecture number 37 where we will talk about a different way of developing a software system that is known as the Agile method. So, before we start, we will first try to understand whatever we have learned so far, what are the issues with those and why Agile development methods often found to be useful in comparison to what methods we have learned in the preceding lectures.

So, if you may recollect, we are dealing with human-computer interfaces. Now, human-computer interfaces are examples of interactive systems. We have seen earlier how a systematic development approach helps us in developing an interactive system software. So, here by the term system we have to remember that we are talking about software development mainly.

So, we require systematic development methods in order to address several concerns that are related to the acceptability of the end product to the customers as well as the end users. One way to do that is to follow software development lifecycles.

(Refer Slide Time: 02:26)



SDLCs

- We have seen two (water-fall, ISLC)
- There are others (spiral, iterative, evolutionary …)
- These are all "traditional" models
  - Stages are mostly the same with variations in execution sequence
  - Equally rigorous with strict requirements at each stage

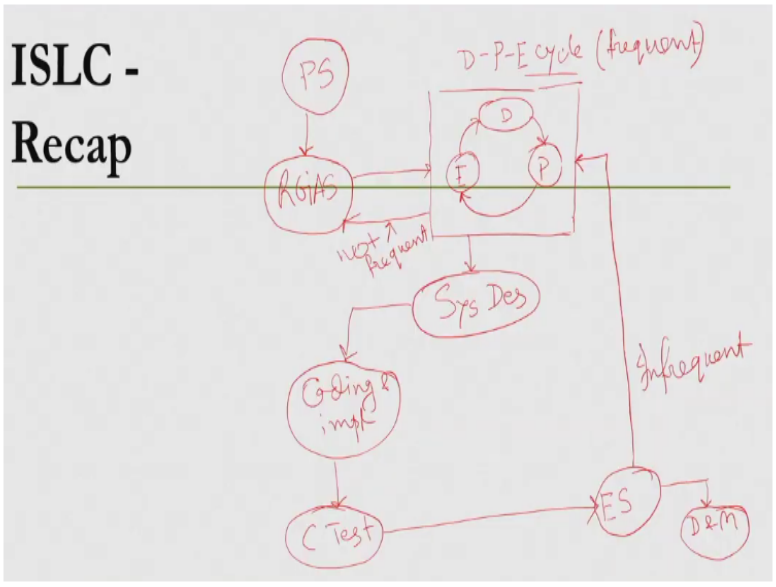
Now, in the early part of this course, in one of the lectures, we have learned about the idea of software development life cycles and also we have seen in details two such lifecycle models, water-fall model and interactive system development lifecycle model. Apart from that, we have also briefly mentioned about several other models that include the spiral model, iterative model, evolutionary model and so on.

So, these are models of software development process, how the process should proceed in a stagewise manner. So each of these models represent the whole development as a combination of stages that are to be performed in a particular sequence. Now, these are all so called traditional models that means in most of the models except probably the spiral model which is a meta level model, most of the model stages are mostly the same with variations in execution sequence.

So, how the stages are to be executed that sequence may vary, but the stages or the core concept of the stages remains the same in almost all the models. Another important characteristic of these so called traditional models is that there is equally rigorous requirements. So, each of the stages need to be executed in a very rigorous manner with strict requirements for almost all of these models.

So, two primary characteristics of these traditional models are that mostly they include the same or similar stages and the stages need to be carried out in a very rigorous manner with strict requirements. Those are the defining characteristics of these traditional models. So, we have seen in details throughout the course the interactive system development lifecycle model.

This is one example of the traditional models. Let us quickly see what are the stages and where the rigor comes from and what we mean by strict requirements. Let us first try to recap the model.
(Refer Slide Time: 04:29)



So, the model comprises of several stages. First is feasibility study stage. So, we will denote it with this acronym PS, feasibility study. This is followed by requirement gathering analysis and specification stage. Now, from here it goes to the combination of substages which together form a cycle, we call it design, prototype, evaluate cycle. So, here we have the design stage, prototyping stage and evaluation of the prototype. And this cycle continues till we arrive at some stable interface design.

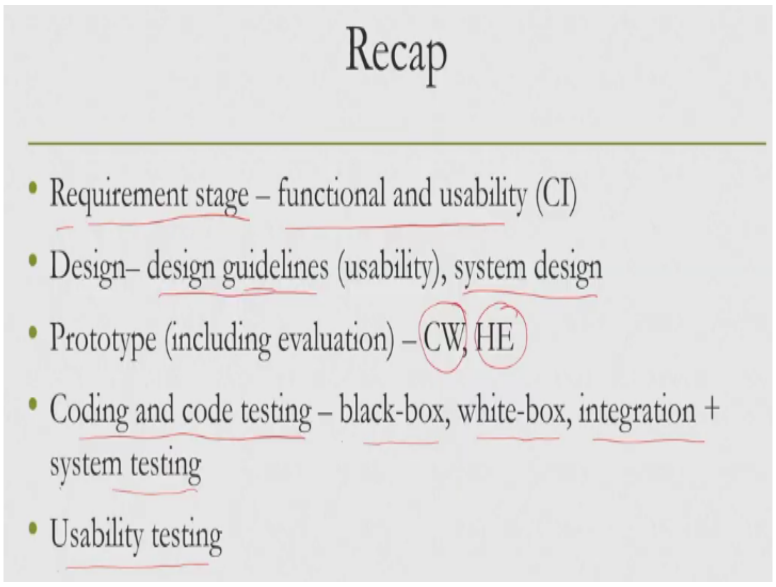
This is followed by a system design stage. So, in this design prototype evaluate stage we are primarily concerned about interface and interaction design to take care of the usability concern, which is one of the primary concerns for interactive system. Once that is done and we arrive at a stable design of the interface and interaction we go to the next phase that is system design where we design the overall structure of the system in terms of modules and submodules.

Now, this is followed by coding and implementation stage. After that, we go for code testing. This is followed by empirical study and this is followed by deployment and maintenance stage. Now, as we have noted earlier, so there may be cycles at this stage. So, from requirements gathering analysis specification, we will go to this design prototype evaluate cycle, but if we find too many problems here then we may come back to refine our requirement gathering process and find out more requirements.

There can be a cycle here as well. So, from empirical study stage if we find some problems, we may go back to this interface design phase, refine our design and then follow the stages again. So, at multiple places there are the presence of cycles, but this cycle should be infrequent, otherwise it will lead to huge cost and time overrun. This cycle may be frequent, actually this is a frequent cycle.

This cycle need not be frequent, it can be between the DEP and the requirement gathering, this cycle need not be frequent, not frequent the cycle. So, we have many cycles in our interactive system development lifecycle that we have covered extensively in the previous lectures. Among them one cycle may be frequent that is designed prototype evaluate cycle, other cycles should not be frequent. So, this is what our interactive system development lifecycle looks.

(Refer Slide Time: 09:29)



Recap

- Requirement stage – functional and usability (CI)
- Design– design guidelines (usability), system design
- Prototype (including evaluation) – CW, HE
- Coding and code testing – black-box, white-box, integration + system testing
- Usability testing

Now, in the stages what we have done? In the requirement gathering analysis and specification stage, we have learned about how to gather functional requirements as well as usability requirements. For usability requirements we can perform techniques such as contextual inquiry, we have covered in details the way to perform contextual inquiry method to gather usability requirements.

So, both functional and usability requirements are gathered, analysed and specified at the end of requirement gathering phase. The specification document is called software requirements specification document or SRS. In the design phase what happens? So, there are two design phases, in one design phase which is part of this design prototype evaluate cycle what we do is we use design guidelines to come up with initial interface and interaction design.

Then we prototype it and get it evaluated with experts for quick evaluation. For quick evaluation, we can follow cognitive walkthrough or heuristic evaluation methods as we have seen earlier. Cognitive walkthrough is a scenario based evaluation method whereas heuristic evaluation is a checklist based evaluation method. Apart from that, we also have system design, there we follow a modular design approach.

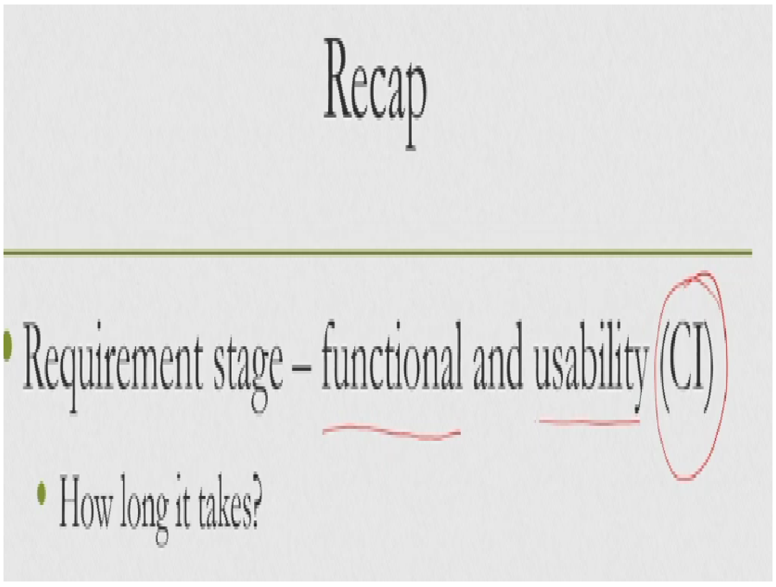
And we can follow either functional design or object-oriented design approaches to design our system and represent it with a corresponding language. For functional design we can use the DFD or data flow diagram and for object-oriented design we can use that UML or unified modelling language to express the design. Coding and code testing, in these stages we follow good coding practices and then we test our system implementation to find out bugs and fix them.

So, different testing methods we have studied namely review-based testing for quick testing of the code, then black box testing or functional testing, white box testing or structural testing, then integration and overall system testing, alpha, beta, acceptance test, etc. plus different performance tests. So, all these things we have covered in details and we have learned in details. In the empirical study stage, what we perform is usability testing.

So, in code testing phase we test the code for executability, in empirical study stage we test the system for usability. We have seen how rigorous the usability testing phase is, it comprises of four stages; research question formulation, variable identification, experiment design and data analysis. So, each of these stages have to be done with great care and in a very planned manner so that at the end we can come to a reliable conclusion about the usability of the product based on the empirical observations.

So, these are the things that we have already covered in details in the previous lectures. Now, each of these stages as we have discussed is a very elaborate and rigorous stage, it needs to be carried out in a very elaborate and rigorous manner which is likely to take a very long time, reasonably long time.

(Refer Slide Time: 13:18)

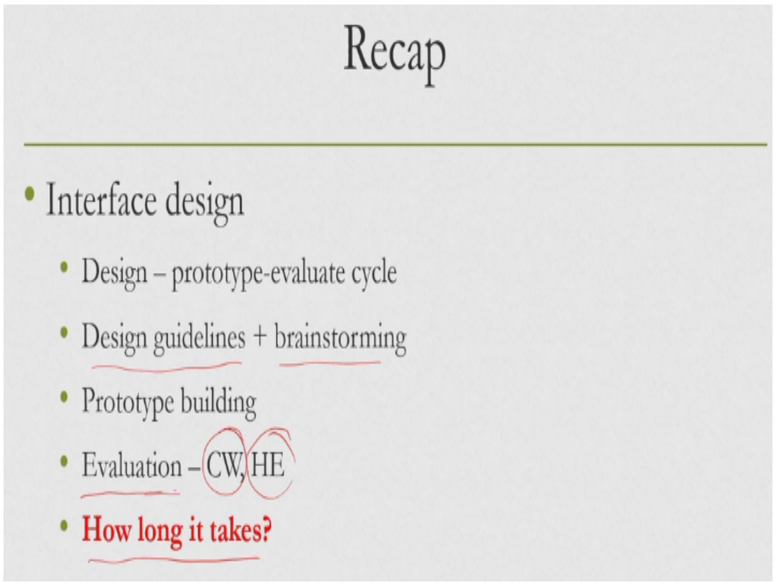

Recap

- Requirement stage – functional and usability (CI)
  - How long it takes?

For example, in the requirements stage, we need to analyse for functional requirements, also we need to analyse for usability requirements which involves application of methods such as contextual inquiry. Now, as we have discussed contextual inquiry is not a very simple method, it cannot be done in few hours or 1 or 2 days.

It requires several days for planning for approaching the participants, for observations, for brainstorming to come to a conclusion about the observations and so on. So, the overall process cannot be completed in a very short span of time, at least 1 week to 2 weeks are required to complete these processes provided everything goes on as per plan.

(Refer Slide Time: 14:02)


Recap

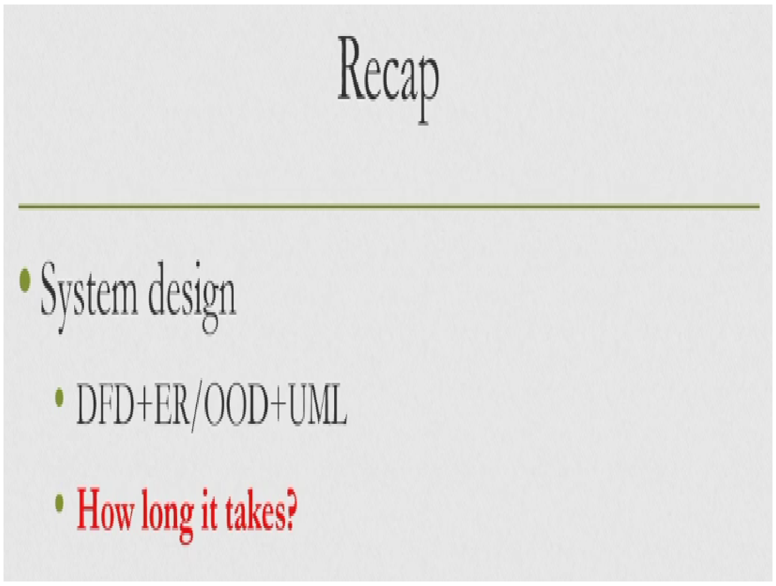
- Interface design
  - Design – prototype-evaluate cycle
  - Design guidelines + brainstorming
  - Prototype building
  - Evaluation – CW, HE
  - How long it takes?

Next, based on the requirement analysis, we can go for design of the interface. Now, it is a cycle, this design cannot be done in one shot as we have seen, so it is a cycle. Design is followed by prototyping followed by quick evaluation, then followed by design refinements and so on. So, when we talk of the cycle, it is not something which can be done quickly. First of all, based on the design guidelines and our own intuition within the design group, we have to brainstorm that takes time.

After brainstorming, we have to come up with a design idea and then we need to create prototypes for that. Finally, we need to go for evaluation using either or both of the methods such as cognitive walkthrough or heuristic evaluation. Now, evaluation also requires forming a team, asking them to perform the evaluation, prepare report, brainstorm, so it has to go through several steps.

Overall, this process again cannot be done in a very quick manner within hours or days, it will take weeks to carry out this process of interface design at least a few weeks are required to come up with a good and stable interface design.

(Refer Slide Time: 15:28)



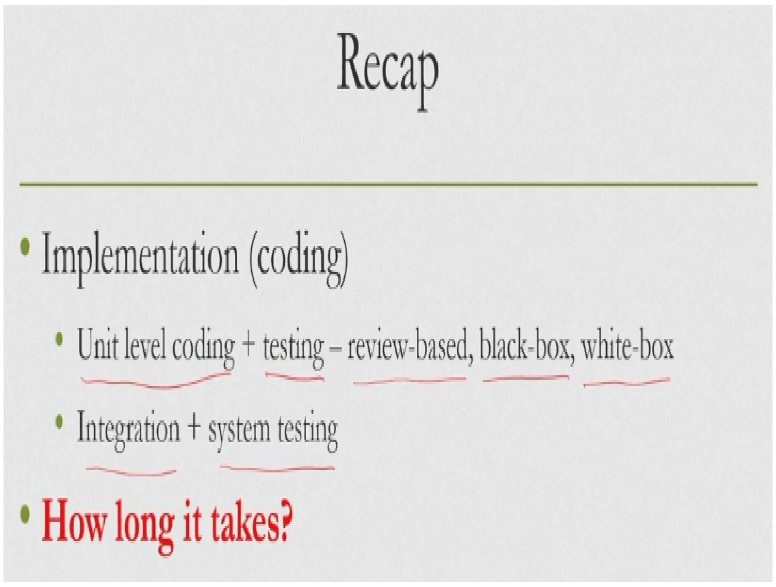
Recap

- System design
  - DFD+ER/OOD+UML
  - How long it takes?

Next comes system design. Now, in system design we have to think of the modular hierarchy, then we have to think of representing our design for that we have to create either DFD diagram if we are following the function-oriented approach or we have to create UML diagrams if we are following object-oriented approach. These diagrams need to be brainstormed to find out any issues with those initial design ideas, then rectifications may be required.

So, again some sort of iteration needs to be performed before we can finalize on system design. So, it is not one shot approach and that iteration again may take weeks. Of course, it can be done in less time compared to the previous stages if we have a very well experienced team and a good amount of effort is put in in this design activity, but generally it cannot be done very quickly, it takes significant amount of time, maybe n weeks.

(Refer Slide Time: 16:38)



Recap

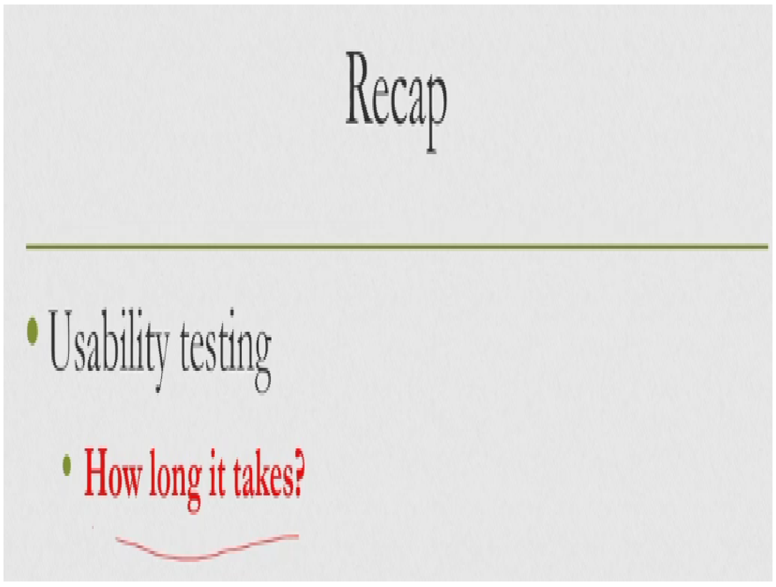
- Implementation (coding)
  - Unit level coding + testing – review-based, black-box, white-box
  - Integration + system testing
- How long it takes?

Coding is the next stage. Coding of course takes up its own time. So, we need to go for unit level coding, then we need to test the code at unit level, review-based testing we need to follow for quick feedback on the code and then we need to refine the code. Then we have to go for rigorous testing methods that include black box testing at functional level and white box testing at structural level.

We also keep on creating documents for all these activities. We also need to go for integration and overall system testing. So, integration of the units or modules that we have put it as well as the overall testing of the system that includes functional testing as well as performance testing. In functional testing, we test for functional support with respect to the SRS document or software requirement specification document.

Whereas in performance testing we test for several non-functional requirements that are listed in the SRS document. This again cannot be done quickly, it requires a good amount of effort from coding team, from testing team and it again is likely to take weeks to complete.

(Refer Slide Time: 18:00)



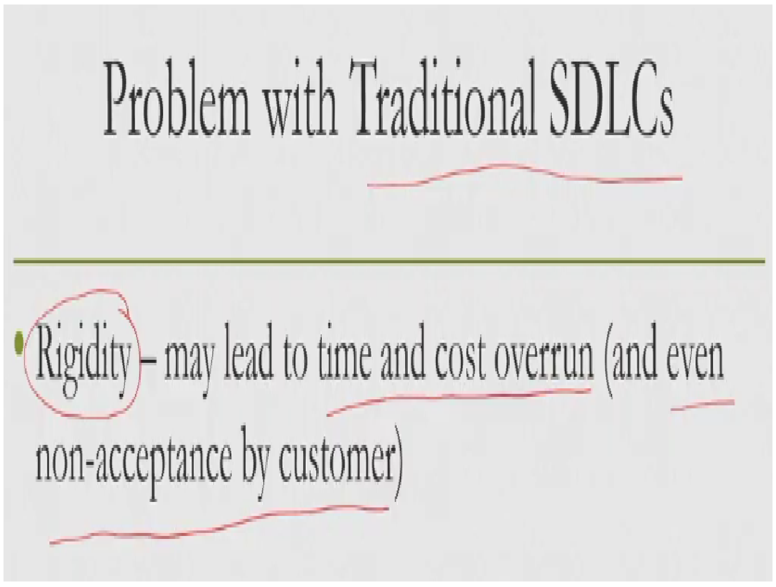
Recap

- Usabilility testing
  - How long it takes?

Usability testing, as we have seen is not very easy either or is a quick process either. How long it can take? We need to first frame research questions that require some brainstorming, Then we need to identify variables and design the experiment that may be done reasonably quick manner, carrying out the actual experiment to collect data takes time depending on the availability and intention of the participants.

Data analysis also takes some time. So overall this can also take weeks, it cannot be done in hours or minutes or in days. However, if participants are readily available and everything is in place, then it can be done in a single week, but in general it may take weeks to carry out.

(Refer Slide Time: 18:58)



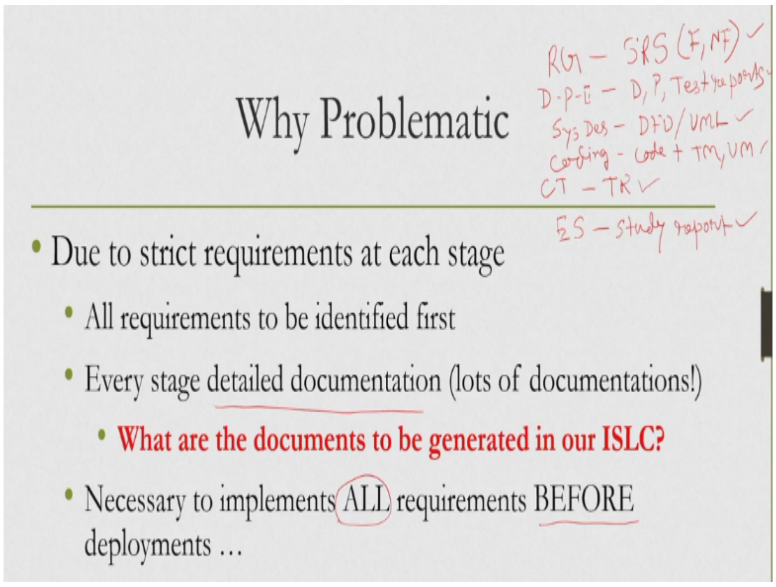
Problem with Traditional SDLCs

- Rigidity – may lead to time and cost overrun (and even non-acceptance by customer)

So, each stage as we can see requires to be carried out with full rigor and that implies that the stages are going to take significant amount of time resulting in a large turnaround time. Now, the problem with this traditional SDLCs as we have just noted is that rigidity. So, whatever is prescribed as part of a system, as part of a stage of the development process has to be done. Deviations are generally not encouraged or allowed and in order to do those things that are prescribed to be done requires time and it also entails cost.

So, this rigidity may lead to time and cost overrun. To comply with the standard practice, time and cost overrun may happen and because of that at the end whatever we get may not be acceptable to the customer. The customer may want them quickly, whereas because of the rigidity of the stages and the overall process, the development team may require more time and cost than permitted by the customer. So, at the end it makes the product unacceptable to the customer.

(Refer Slide Time: 20:20)



Why Problematic

- Due to strict requirements at each stage
  - All requirements to be identified first
  - Every stage detailed documentation (lots of documentations!)
    - What are the documents to be generated in our ISLC?
  - Necessary to implements ALL requirements BEFORE deployments …

RG – SRS (F, NF) ✓
D-P-E – D, P, Test reports ✓
Sys Des – DFD/UML ✓
Coding – Code + TM, UM ✓
CT – TR ✓
ES – study report ✓

Now, why this problem comes? Due to the strict requirements at each stage, we have just seen what are those requirements, in the requirement gathering phase we have to identify all requirements, so later on we should not focus on requirement gathering. So, first we have to start with requirement gathering and for that we have to ensure that all the major requirements are gathered in the requirement gathering stage itself, so that requires time.

In every stage we need to create detailed documentation that entail lots of documentations. So, what are the documents that need to be generated for our ISLC that is what we have already seen in previous lectures. And also, it is necessary to implement all the requirements before the deployment. So, whatever requirements we have identified in the requirement gathering stage needs to be implemented, tested, finalized before we go for deployment.

So, first of all we need to identify all possible requirements that itself takes time and we also need to implement and test all those requirements before we go for deployment that takes time plus we have this issue of detailed documentation at each stage, which also takes a considerable amount of effort and time. So, what are those documents that we need to develop, let us try to list. In the requirement gathering stage we have SRS which contains both functional and non-functional requirements.

So, we have already seen how these documents look like, how it can be created, what are the sections that are part of the document. In the design, prototype, evaluate cycle we have designed document for interface design, then the prototypes as documents for prototypes and test reports after each evaluation. In the system design stage, we have either DFD or UML documents created to represent the system design.

In the coding stage, we have the code with comments plus some code related documents such as technical manuals and user manuals, these can be created at this coding stage. In that code testing stage, various test reports that includes review-based test report, black box test report, white box test reports are to be generated. Now, in the reports of course you have to describe the method and other details in each of these reports.

For example, in black box testing report you have to mention the equivalence classes, the functions, equivalence classes for each function, test cases that you have used. All these things have to be mentioned as we have discussed in case studies. Empirical study also involves creation of a detailed study report incorporating the research questions, the hypothesis, the experimental procedure, method, setup, design, then detailed statistical significance testing method and results.

So, all these things have to be fairly and properly documented after carrying out the corresponding stage. So, just to repeat we require several documents starting with SRS in the requirement gathering analysis and specification stage as outcome of the stage. After the interface and interaction design stage we produce design documents, prototype documents and test reports.

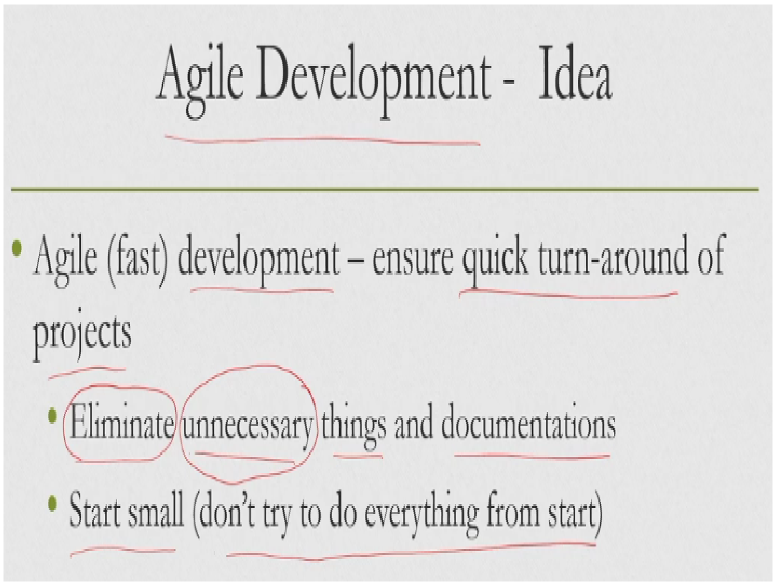
After the system design phase, we produce DFD or UML documents to represent the design of the system. After the coding phase, the code itself is the document with comments of course plus some technical manual and user manuals for better understanding of the code. After the code testing phase, we produce test reports, several test reports can be produced depending on the amounts of tests done. So, if we are following review-based testing first, then review-based test report.

Then if we are performing more rigorous tests such as the black box test and white box test, then for each test the corresponding reports need to be produced with all the details of the testing process. Finally, for empirical study we need to produce another report detailing everything about the study including the research questions, the hypothesis, the experiment design, method, setup and statistical significance test details. So, all these documents we have to produce.

And somebody has to check the documents as well requiring additional manpower, additional time, additional cost. So, you can see how rigorous and how time consuming this process is which involves rigorous execution of the stages as well as rigorously creating the documents. So, the result of all these things that is the strict requirements and documentation requirements imply that if we follow traditional system development method, then it is likely to take long amount of time.

Also it is going to entail good amount of cost because of maintaining the sanctity of the process, the strict requirements. In order to address this issue because sometimes this time and cost factors become very important, sometimes it is the objective of the developers to deliver a product quickly even if that requires compromising with the development process to some extent, but the objective is to quickly deliver the product in some acceptable condition that brings us to this concept of agile development.

(Refer Slide Time: 27:15)



Agile Development - Idea

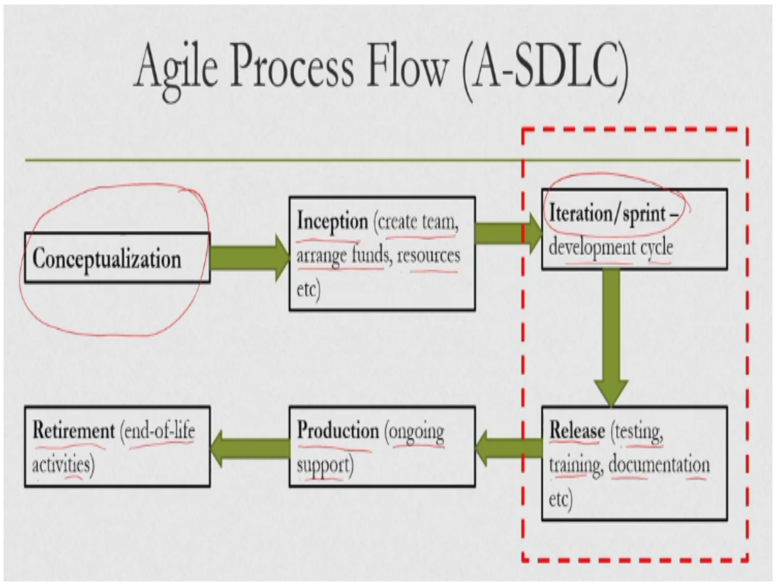
- Agile (fast) development – ensure quick turn-around of projects
  - Eliminate unnecessary things and documentations
  - Start small (don't try to do everything from start)

As the name suggests, agile which means fast, agile development process implies a quick turnaround time of the end product. So, quick turnaround of the project that is what is the primary objective of this type of developmental activities that is called agile development process. In order to be agile, what this process does is try to eliminate unnecessary things and documentation that may deem unnecessary at certain stages of the development.

So, although you are using the term unnecessary, which is generally used to differentiate between traditional and agile development processes, it may be noted that this term is too strict. The stages and the requirements need not be unnecessary if we look at it very holistic point of view. However, in agile it is considered to be unnecessary if some requirements are not relevant at particular stages for quick turnaround time. We will see what it means.

Other key aspect of agile development is that it relies on the philosophy of start small that means do not try to do everything from the very beginning unlike the traditional methods where we try to first gather all requirements in the requirement gathering stage, we try to do all development in the development stage, so everything we try to do in a complex manner whereas in agile that is not the philosophy. In agile, it is said that try to start with small things.

(Refer Slide Time: 29:13)

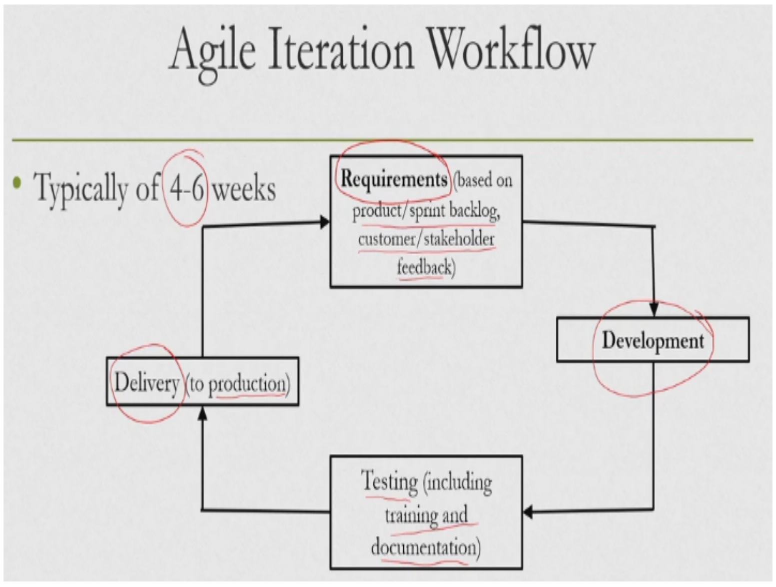

Agile Process Flow (A-SDLC)
Conceptualization
Inception (create team, arrange funds, resources etc)
Iteration/sprint – development cycle
Release (testing, training, documentation etc)
Production (ongoing support)
Retirement (end-of-life activities)

So, how the process flow looks. You will find similarity with traditional SDLC of course in terms of the basic concepts, but there are differences. So, it starts with conceptualization of the overall process. This is followed by inception stage which includes creation of a team for development purpose, arranging for funds, resources, etc. Now this is followed by an interesting concept called iteration or sprint which is the development cycle.

We shall soon see what is there in the cycle. Once the cycle is over, it goes to the release stage which involves testing, training and necessary documentation. This is followed by production stage which includes ongoing support as well followed by retirement stage of the product that is end of life activities. So, these are broadly the stages that are part of any generic agile development process.

So, it starts with the conceptualization stage, ends at the retirement stage, in between there is inception after conceptualization followed by iteration popularly known as sprint followed by release followed by production. So, these stages from the core of any agile development process.

(Refer Slide Time: 30:39)


Agile Iteration Workflow
• Typically of 4-6 weeks
Requirements (based on product/sprint backlog, customer/stakeholder feedback)
Development
Testing (including training and documentation)
Delivery (to production)

Now, the iteration or sprint is the heart of the process, there actually the development takes place. Now, typically in an agile development environment, the iteration should not take more than 4 to 6 weeks. Now, of course since we are imposing some time constraint that means we are cutting down on some strict requirements that are part of the development process, if we follow a traditional approach. So, in this iteration what happens?

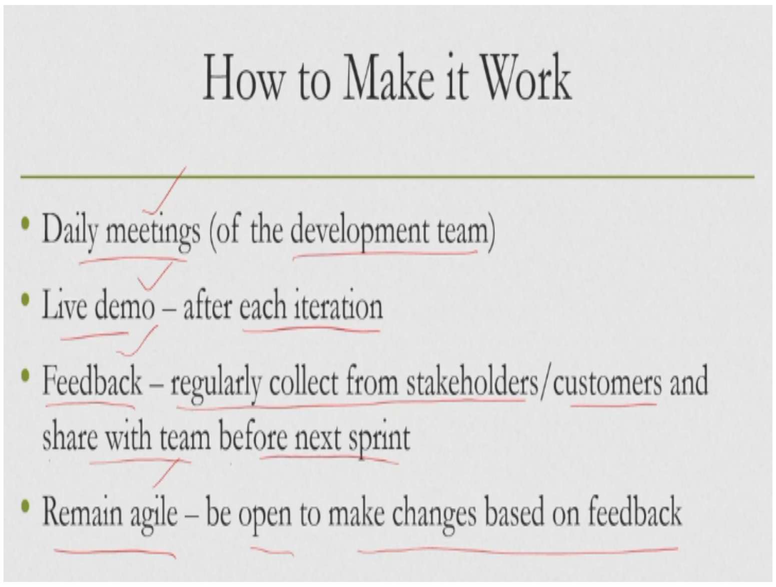
First there is some requirement gathering, so that is part of the requirements stage. In this stage what happens is that based on the product or backlog from the previous iterations as well as customer or stakeholders or both feedbacks, some quick requirement gathering takes place. This is followed by the actual development process, development of the system based on the requirements that is the coding.

This is followed by testing, testing of the things that have been developed which include training and documentation and this is followed by delivery to the production stage and this goes on till we finish delivering all the features in the product that are supposed to be delivered. So, how it is different from the other development stages in traditional methods? In traditional methods, what we try to do?

We try to gather all requirements at the beginning and try to develop all requirements in the system before we deliver. Here what we do? We start small, we try to gather some requirements at the beginning, develop the system, test it for those requirements only and release it and then we continue for other requirements that is what this iteration means. So, we do not try to gather all requirements at the beginning.

We do it in an incremental manner, in an iterative manner and at the same time we keep on releasing updated products. So, in that way it is expected that the customers get something after a small amount of time 4 to 6 weeks rather than waiting for the whole cycle to complete and all the requirements are identified and developed and tested before it is given to the customer.

(Refer Slide Time: 33:22)



How to Make it Work

- Daily meetings (of the development team)
- Live demo – after each iteration
- Feedback – regularly collect from stakeholders/customers and share with team before next sprint
- Remain agile – be open to make changes based on feedback

So, how to make it work? How we can ensure that within 4 to 6 weeks of time we can deliver a product starting with requirement gathering to coding to testing and finally releasing to the customer. For that we; we means the project managers, require to conduct daily meetings of the development team. These are some of the traditional steps that are followed to ensure that the agile development works. Then live demo after each iteration that is required.

This is followed by feedback. So, we need to regularly collect from the stakeholders or the customers or even the end users' feedback and share with the team before next sprint. So that is what I said requirements are not collected at the beginning, all the requirements, and requirements collection also depends on feedback. So once something is developed, then based on the feedback we can refine the requirements and then go for next iteration or next sprint.

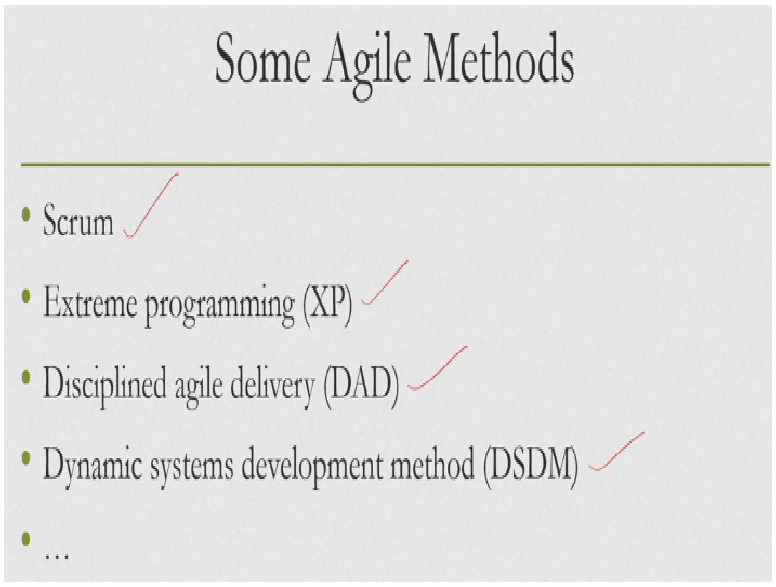
So, these regular collections of feedback should be part of the overall process as it helps us to carry out the next sprint quickly just after the finishing of the earlier sprint. Remain agile that means the team should be open to make changes based on feedback. So, it is not that now I have developed something and whatever I have developed is very good, take it or leave it, instead what the philosophy should be is that okay I have developed something, more can be developed depending on feedback.

So, whenever the feedback comes, I should be ready for development with another sprint that should be the philosophy imbibed by the team members. So, these are some things that we need to follow to ensure that we abide by the agile development process. It involves regular meetings regarding the sprint, live demo of the product that has been developed and tested at the end of each iteration or sprint.

Regular feedback from stakeholders or customers or both for getting more information about the product as well as to prepare for next sprint if required and remain agile that means, we should have this mentality that whatever we have developed can be improved and we should be able to improve it quickly in an agile manner depending on the feedback. These things primarily determine whether the method that you are following can be agile or not.

Sometimes we have this mental blockage of accepting the fact that whatever we have developed lacks something and we need to develop it again, so that needs to be removed. So, agile development method is a very generic concept. Now there are specific technologies developed, specific process or methods developed to implement this agile development concept.

(Refer Slide Time: 37:03)



Some Agile Methods

- Scrum ✓
- Extreme programming (XP) ✓
- Disciplined agile delivery (DAD) ✓
- Dynamic systems development method (DSDM) ✓
- …

Some are very popular, namely the scrum which is an agile method, extreme programming or XP another agile method very popular, disciplined agile delivery or DAD, dynamic systems development method or DSDM, etc. So, some of these are quite popular others are not so popular. So, if you are following a scrum method that means you are following an agile method that is the significance of these terms.

So, that is in brief what is agile development method. Just to recap, we have learned traditional software development methods. Now, these methods come with lots of stages. Each of these stages needs to be carried out in a very rigorous manner and at the end of each stage we need to produce rigorously extensive documentation, both these activities require us to invest heavily on manpower and increases the cost of the overall project in terms of time as well as money.

Sometimes that may make things unacceptable to the end users. In order to avoid that, this agile development philosophy came into being. It is not very new, it is quite old now, but many a times this methodology is followed. Here, the idea is that we do not try to do everything at every stage, instead we start small do something, deliver it and then we go for delivering something more unlike traditional methods where we try to deliver everything at the end.

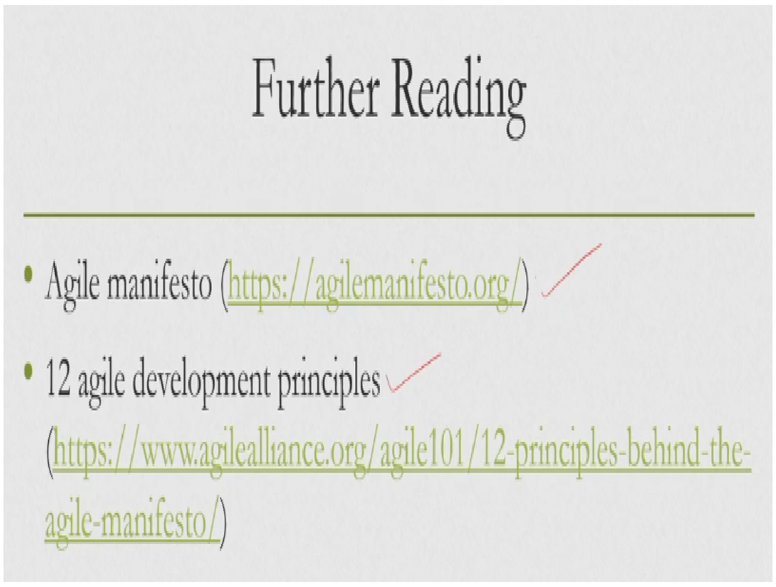
So, in between deliveries not considered whereas in agile method we keep on delivering to the client things in an incremental manner that is the primary difference. In order to do that several concepts have been introduced including the concept of iteration or sprint. Each sprint takes about 4 to 6 weeks' time and in each sprint what is done is some requirements are identified, then it is developed into a system.

The system is tested and the tested system is delivered. Now, the requirements need not be all the requirements that are there, only a few. And once the sprint is over based on the feedback, we can go for another sprint to add more features or more requirements into the system. So, after 4 to 6 weeks, it is expected that some products can be delivered to the clients so that clients are also happy that is the primary idea.

And in order to make this agile method work several things need to be taken care of including regular meetings between the team members, among the team members, then regular feedback, open mind that whenever feedback comes based on that we should be ready to change and live demo after each sprint. So, there are some popular methods namely Scrum, XP, DAD, etc., which are used to follow the agile development concept.

With that, we have come to the end of this lecture. I hope you have enjoyed the material that we have covered, although we covered it in very brief this is a very important concept, very interesting also and it is preferable if you try to understand in more details about the concepts that we have briefly introduced.

(Refer Slide Time: 41:05)



Further Reading

- Agile manifesto (https://agilemanifesto.org/)
- 12 agile development principles (https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/)

For more details on these concepts, you can refer to these materials, they are online materials. The primary document is Agile manifesto. Also, you can follow these 12 agile development principles that you can find in this link that is shown on the screen. So, apart from that if you can get access to any book on agile development method, you are also advised to follow that book to know more details about the method.

So, whatever method we learned that is the interactive system development lifecycle and how it is different from agile approach that distinction by now should have been clear to you. One thing we should keep in mind is that although I talked of agile method, but it does not mean that the other methods, the traditional methods are useless. Sometimes the time that is required to implement a fully traditional method may not be acceptable to the clients that time we may have to follow an agile method.

So, we should be aware of the existence of such methods, however that does not mean that we should totally discard the traditional development method. Knowledge of both is good, but it is not mutually exclusive. So, we should not think that one method is sufficient, other method we need not know. With that I would like to end this lecture. Hope you have enjoyed the content and looking forward to meet you all in the next lecture. Thank you and goodbye.