Software Engineering
Prof. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

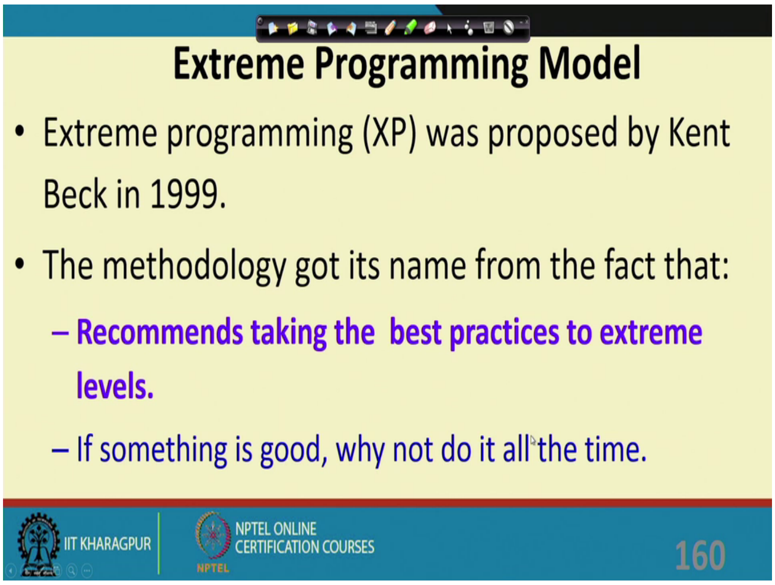
Lecture - 13
Extreme Programming and Scrum

Welcome to this lecture. In the last lecture we looked at the agile development. In the changing software projects, we had seen that over the years a lot of changes have occurred to the software projects themselves. From multiyear projects this have become couple of weeks and also lot of reuse and customization is being made and from product development, many projects are now service oriented projects.

In this context the agile model has become extremely popular. In any software development organization you go see that large number of projects they are following the agile development practices. And, we had said that agile is actually an umbrella term, there are certain characteristics that all agile development projects have to follow, but then there are several methodologies which come under the agile umbrella. And 2 prominent methodologies here are the extreme programming and the scrum.

In this lecture we look at these 2 agile methodologies the extreme programming and a scrub let us proceed. We will first look at the extreme programming also called as a XP.

(Refer Slide Time: 02:08)



Extreme Programming Model

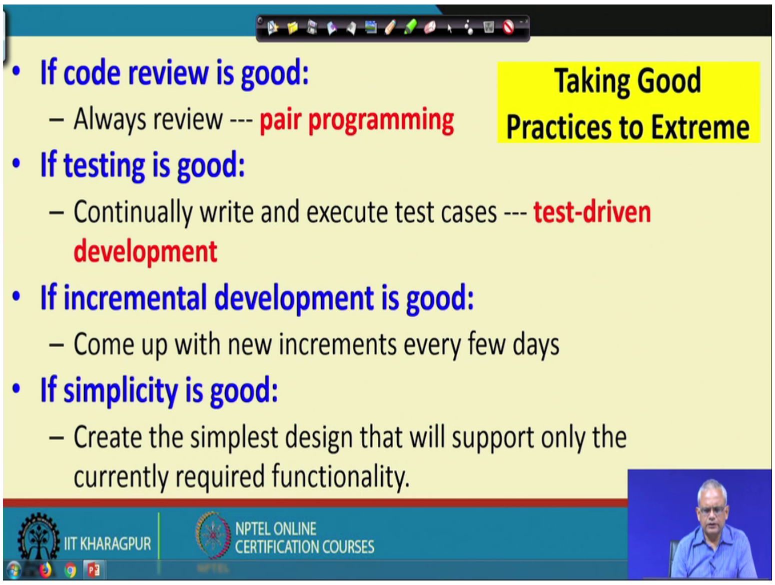
- Extreme programming (XP) was proposed by Kent Beck in 1999.
- The methodology got its name from the fact that:
  - Recommends taking the best practices to extreme levels.
  - If something is good, why not do it all the time.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL
160

It was proposed by Kent Beck in 1999. The name extreme here, implies that the best practices are taken to the extreme level, whatever works will in some projects why not put it to the maximum use. So, that is the main focus of this model and it has all other characteristics of a agile model.

So, the principle here is that if something works and is good why not maximize it is use.

(Refer Slide Time: 03:08)



Taking Good Practices to Extreme

- If code review is good:
  - Always review --- pair programming
- If testing is good:
  - Continually write and execute test cases --- test-driven development
- If incremental development is good:
  - Come up with new increments every few days
- If simplicity is good:
  - Create the simplest design that will support only the currently required functionality.

Now, let us see what are some of the good practices? And how these are taken to the extreme level? Research results suggests that code review is a very good practice, code review is a much better than testing, code review can eliminate bugs in the software most cost effectively, and even can eliminate bugs which cannot be detected by testing or is very difficult to detect by testing.

So, it has been accepted by all developers, that code review is a good practice. The extreme programming proposes to take this code review to extreme level by pair programming, but then what is pair programming? In pair programming the code is written by 2 programmers on one desk. So, on one computer there are 2 programmers, one programmer writes the code while the other programmer reviews that. And, then the interchange every half an hour or so one programmer writes the other reviews then the switch their place. So, 2 programmers have understanding of the code they put their mind together and also the review any mistakes and so on is pointed out by the other programmer.

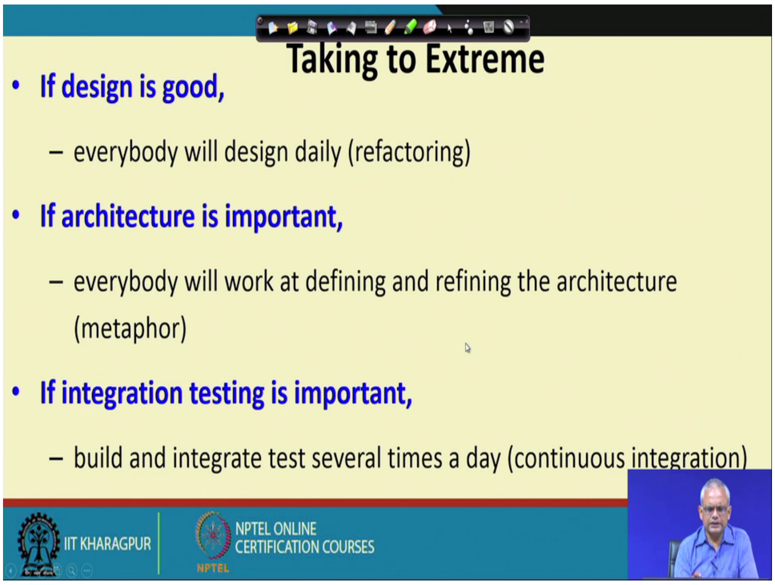
The other good practice is testing is good it makes software more reliable, to make testing to take testing to the extreme level the extreme programming suggests to write the test cases continuously. So, that is called as test driven development in test driven development, even before the code is written the test is written the test cases are written.

And, after the test cases are written, the code is written, and each time the code is written to some extent the test cases are run to see if the code passes that, otherwise the code is modified. Incremental development is good it said that it eliminates lot of problems of the waterfall model. And therefore, extreme programming says come up with a new increment every few days.

Simplicity is good because it produces less bugs easy to maintain and therefore, the extreme programming says take the simplicity to the extreme level and for that it says that only focus on what is needed now and do the simplest design for that. Do not worry about what will be needed after 10 years that may never be needed, do not worry about that at all do the best design under the present circumstance.

So, pair programming test driven development, incremental development in a few every few days' simplicity of the design these are important characteristics of the extreme program.

(Refer Slide Time: 07:28)



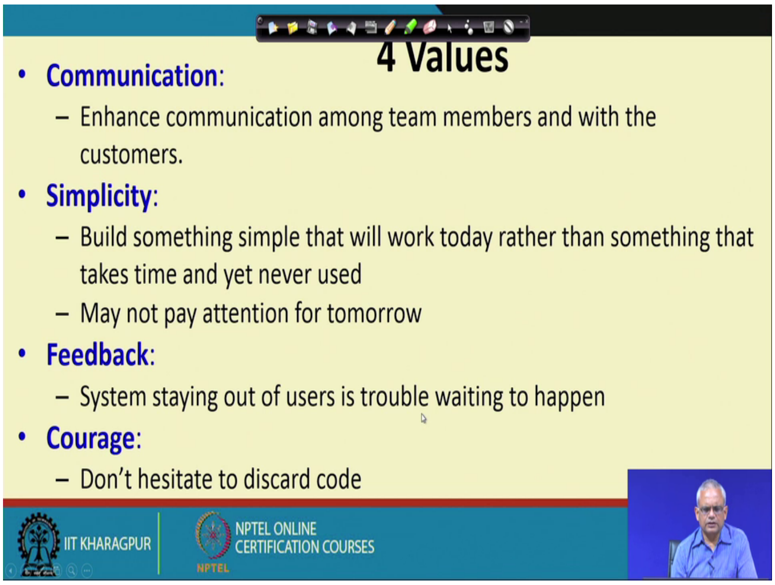
Taking to Extreme

- If design is good,
  - everybody will design daily (refactoring)
- If architecture is important,
  - everybody will work at defining and refining the architecture (metaphor)
- If integration testing is important,
  - build and integrate test several times a day (continuous integration)

The other things that are taken to the extreme are designing is good. So, do designing after the code works. Everybody will design daily they will refactor once the code works, but design into it that is called as a refactoring architecture overall design is important. So, metaphor that is the agile principle everybody will define and refine the metaphor.

Integration testing is a important we had seen that in the waterfall model. The delay starts from integration testing the schedule slippage initially starts at the integration testing and becomes worse after the integration testing, but here in the extreme programming. We must build every day and whatever we build same day we must integrate all together and integrate several times a day continuous integration. So, that later surprises are not there.

(Refer Slide Time: 08:47)

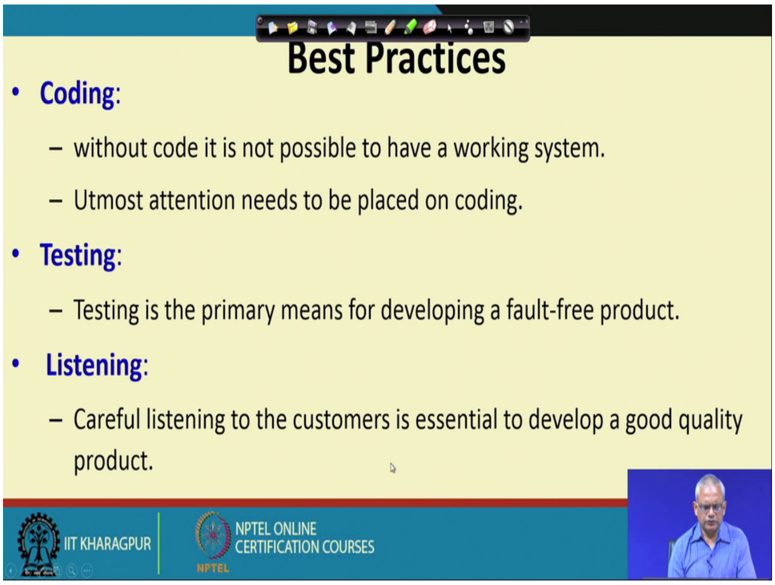


4 Values

- **Communication:**
  - Enhance communication among team members and with the customers.
- **Simplicity:**
  - Build something simple that will work today rather than something that takes time and yet never used
  - May not pay attention for tomorrow
- **Feedback:**
  - System staying out of users is trouble waiting to happen
- **Courage:**
  - Don't hesitate to discard code

Just like any other agile model, enhanced communication among team members, face to face communication and also with the customer. Simplicity; builds something simple that will work rather than make it extremely complicated about something which never be required. Take continuous customer feedback and the customer can be part of the team and if the code structure has become bad discard it write new code.

(Refer Slide Time: 09:29)



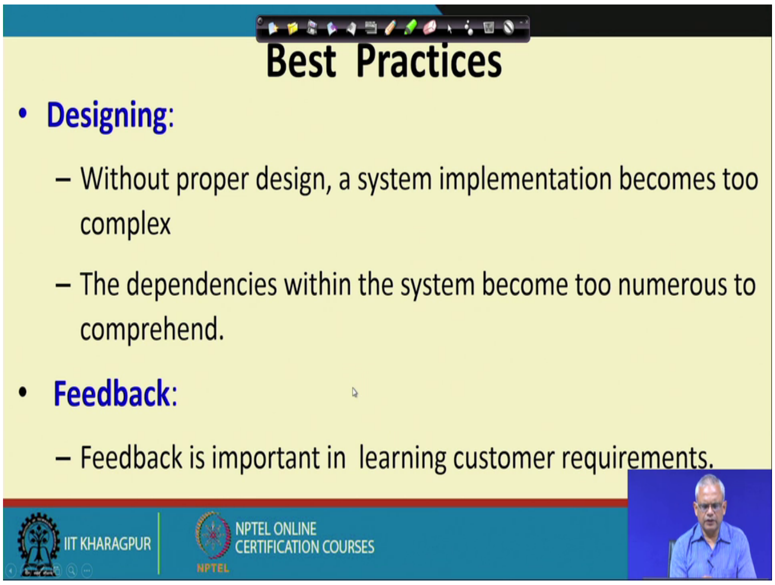
Best Practices

- **Coding**:
  - without code it is not possible to have a working system.
  - Utmost attention needs to be placed on coding.
- **Testing**:
  - Testing is the primary means for developing a fault-free product.
- **Listening**:
  - Careful listening to the customers is essential to develop a good quality product.

Coding is a important a place in extreme programming at most attention need to be placed on coding, testing is important and also listening to the customer is essential to develop good quality product.

So, listen very carefully to the customer feedback.

(Refer Slide Time: 09:55)

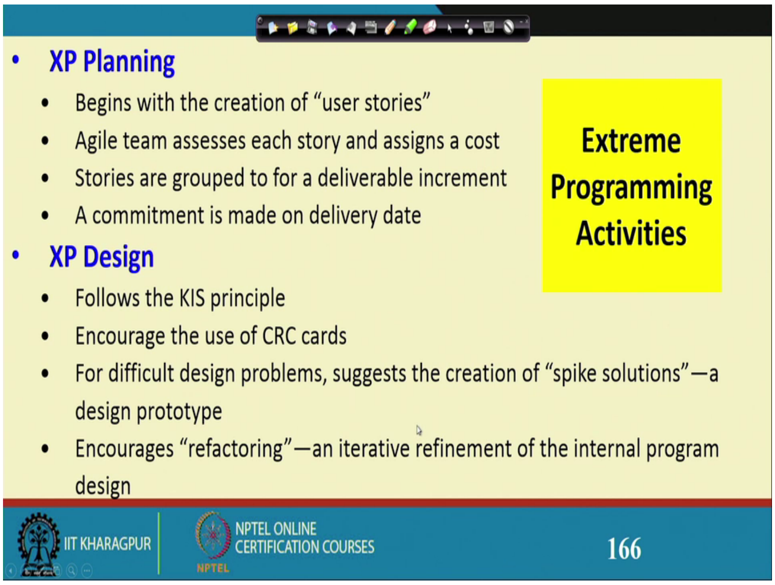


Best Practices

- **Designing**:
  - Without proper design, a system implementation becomes too complex
  - The dependencies within the system become too numerous to comprehend.
- **Feedback**:
  - Feedback is important in learning customer requirements.

Designing after the code works put design into it called as a refactoring; feedback is important learning the exact customer requirement.

(Refer Slide Time: 10:07)



- **XP Planning**
  - Begins with the creation of "user stories"
  - Agile team assesses each story and assigns a cost
  - Stories are grouped to for a deliverable increment
  - A commitment is made on delivery date
- **XP Design**
  - Follows the KIS principle
  - Encourage the use of CRC cards
  - For difficult design problems, suggests the creation of "spike solutions"—a design prototype
  - Encourages "refactoring"—an iterative refinement of the internal program design

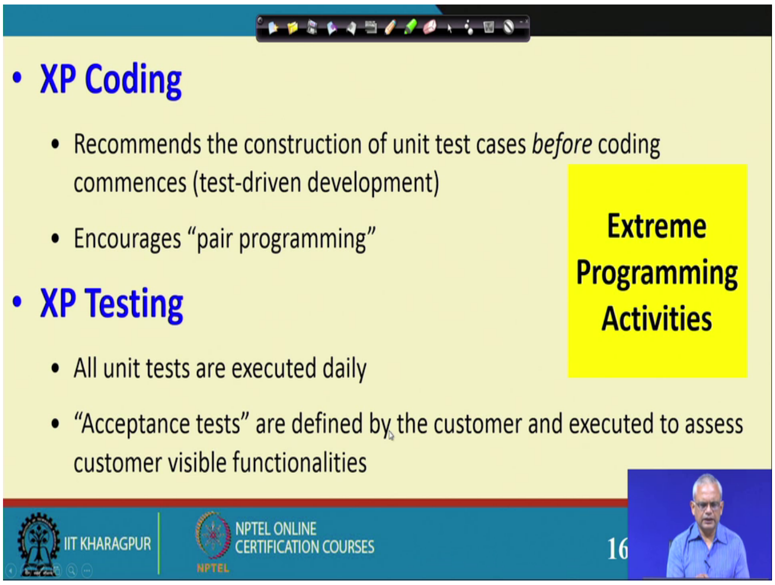
Extreme Programming Activities

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | 166

The planning are only a short term and each story or requirement is assigned a cost. And, these are grouped into deliverable increments. Design is keep it simple principle use of CRC cards that is class responsibility collaboration, which is pioneered by Kent Beck we will see the technique as we look into the design aspects.

And, whenever there are confusion about which is the best way to go create a spike solution alternate between, choose between alternative solution by designing prototypes and after a code works refactor and put design into it.

(Refer Slide Time: 11:10)



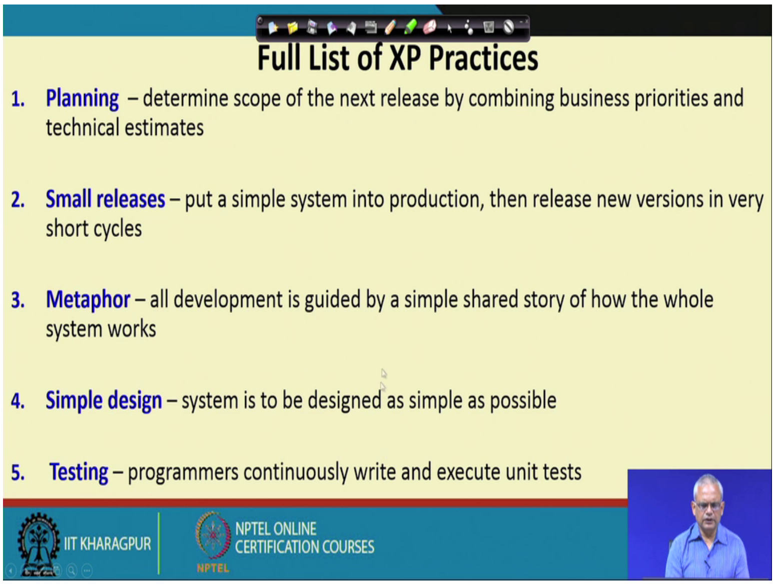
- **XP Coding**
  - Recommends the construction of unit test cases *before* coding commences (test-driven development)
  - Encourages "pair programming"
- **XP Testing**
  - All unit tests are executed daily
  - "Acceptance tests" are defined by the customer and executed to assess customer visible functionalities

Extreme Programming Activities

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | 16

The coding is by test driven development first write the test cases before any code is written. And, then write the code until it passes the test cases written pair programming is encouraged all software that is developed that tested daily and acceptance tests are defined and are executed.

(Refer Slide Time: 11:46)

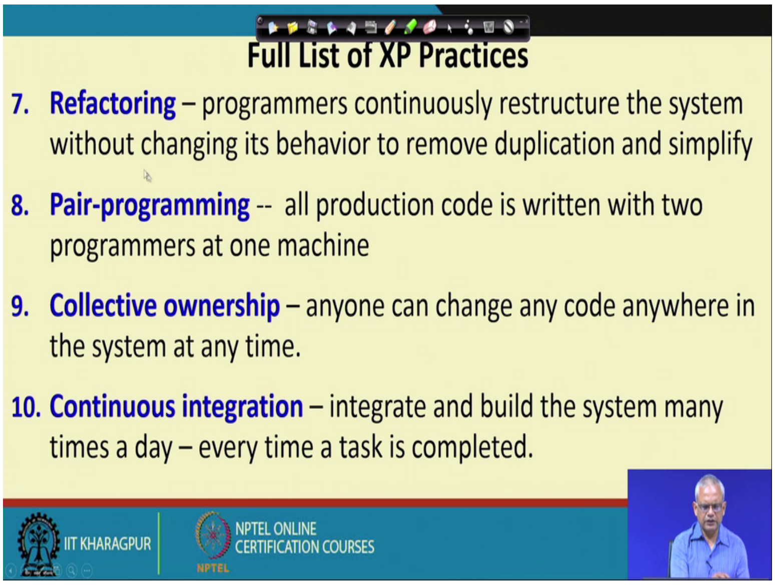


Full List of XP Practices

1. **Planning** – determine scope of the next release by combining business priorities and technical estimates
2. **Small releases** – put a simple system into production, then release new versions in very short cycles
3. **Metaphor** – all development is guided by a simple shared story of how the whole system works
4. **Simple design** – system is to be designed as simple as possible
5. **Testing** – programmers continuously write and execute unit tests

These are the full list of the extreme programming practices, planning, only short term plans are made, small religious, metaphor, simple design, testing, continuously write and execute test cases.

(Refer Slide Time: 12:05)

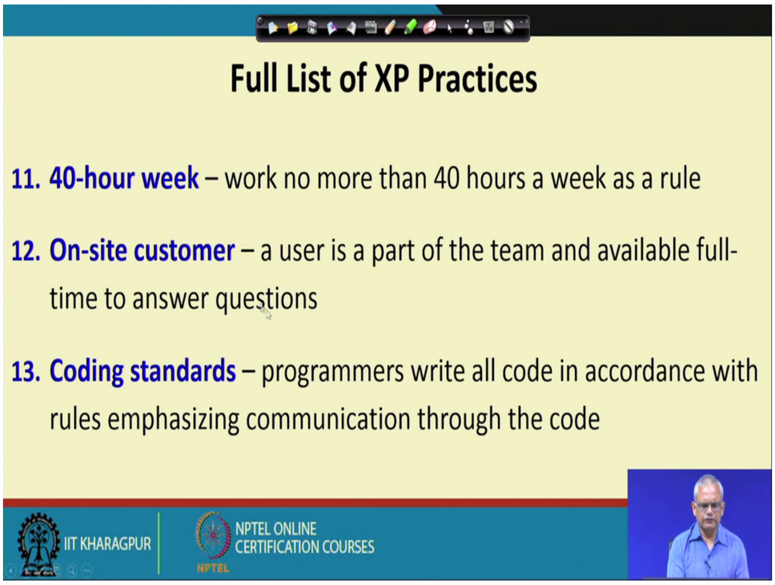


Full List of XP Practices

7. **Refactoring** – programmers continuously restructure the system without changing its behavior to remove duplication and simplify
8. **Pair-programming** -- all production code is written with two programmers at one machine
9. **Collective ownership** – anyone can change any code anywhere in the system at any time.
10. **Continuous integration** – integrate and build the system many times a day – every time a task is completed.

Refactoring, Pair-programming, collective ownership not only that 2 programmers they write a piece of code, but any programmer should be able to modify any other programmer's code. Continuous integration every day whatever code is written has to be integrated with the existing code.

(Refer Slide Time: 12:29)



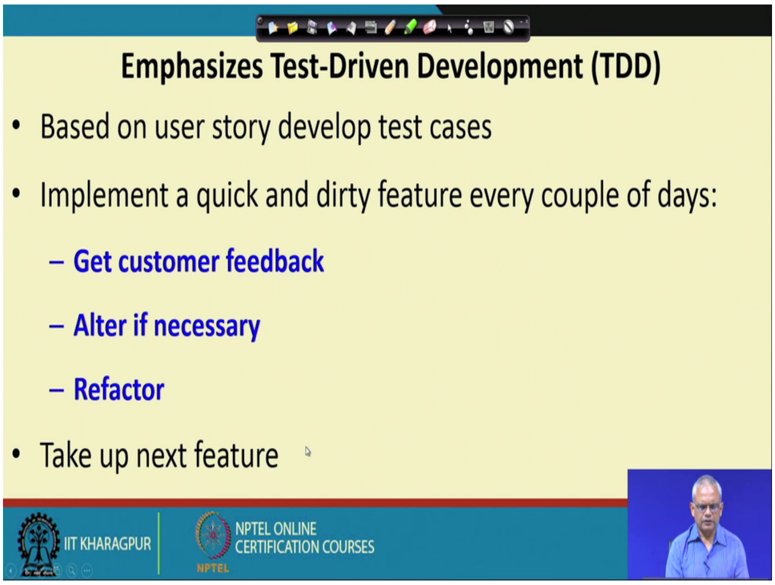
Full List of XP Practices

11. **40-hour week** – work no more than 40 hours a week as a rule
12. **On-site customer** – a user is a part of the team and available full-time to answer questions
13. **Coding standards** – programmers write all code in accordance with rules emphasizing communication through the code

40-hours week customer must be part of the team and coding standards to be followed, as you can see that many of these are actually philosophical and little bit sketchy.

And therefore, good quality manpower is needed for extreme programming or any other agile projects.

(Refer Slide Time: 13:07)

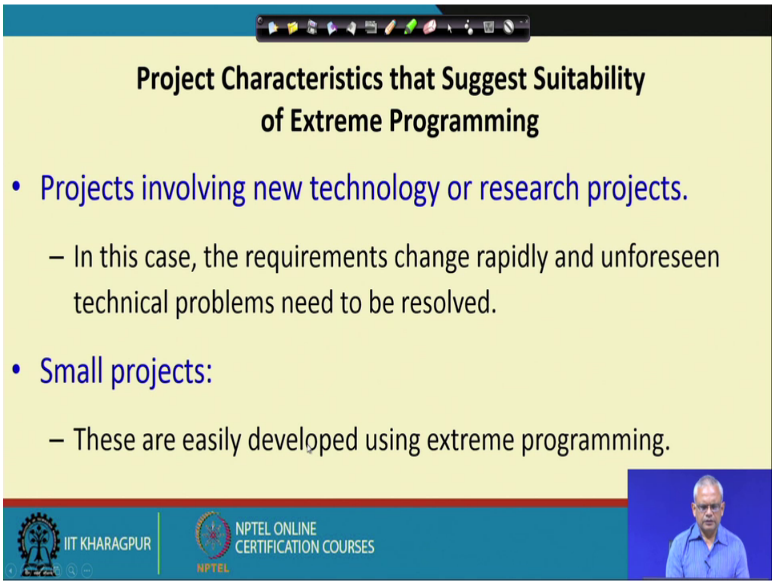


Emphasizes Test-Driven Development (TDD)

- Based on user story develop test cases
- Implement a quick and dirty feature every couple of days:
  - Get customer feedback
  - Alter if necessary
  - Refactor
- Take up next feature

It emphasizes test driven development any increment starts with a user story the first thing is to develop test cases based on the user story. And, then develop the software get the customer feedback based on the customer feedback alter necessary and then put design into the code, make the code refine it into good quality code and then take up the next feature.

(Refer Slide Time: 13:44)

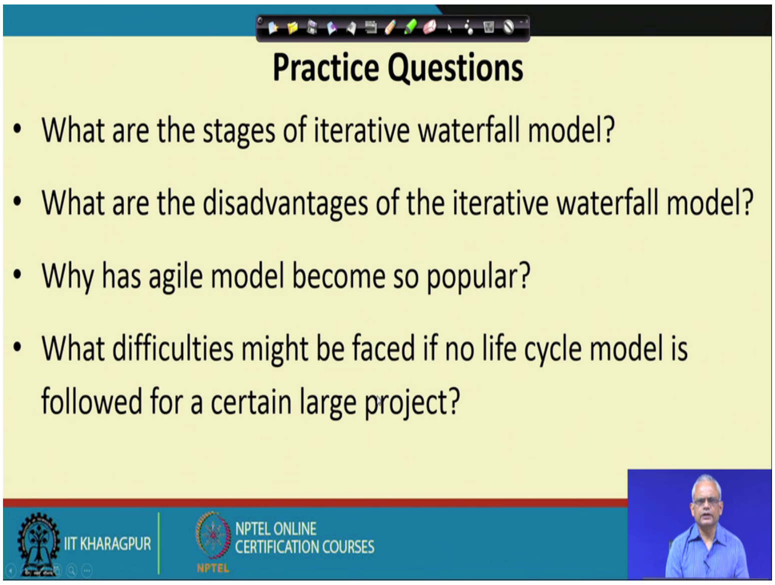


Project Characteristics that Suggest Suitability of Extreme Programming

- Projects involving new technology or research projects.
  - In this case, the requirements change rapidly and unforeseen technical problems need to be resolved.
- Small projects:
  - These are easily developed using extreme programming.

Extreme programming is successful in small projects and also the projects which are of challenging nature.

(Refer Slide Time: 14:00)



Practice Questions

- What are the stages of iterative waterfall model?
- What are the disadvantages of the iterative waterfall model?
- Why has agile model become so popular?
- What difficulties might be faced if no life cycle model is followed for a certain large project?

Now, having looked at several development models including some of the agile models let us try to check our understanding with few questions. Can you identify what are the stages of the waterfall model? It is a very very basic question feasibility study requirement analysis and specification design, coding, unit testing, testing and maintenance, but then how are the feedback paths organized between them?

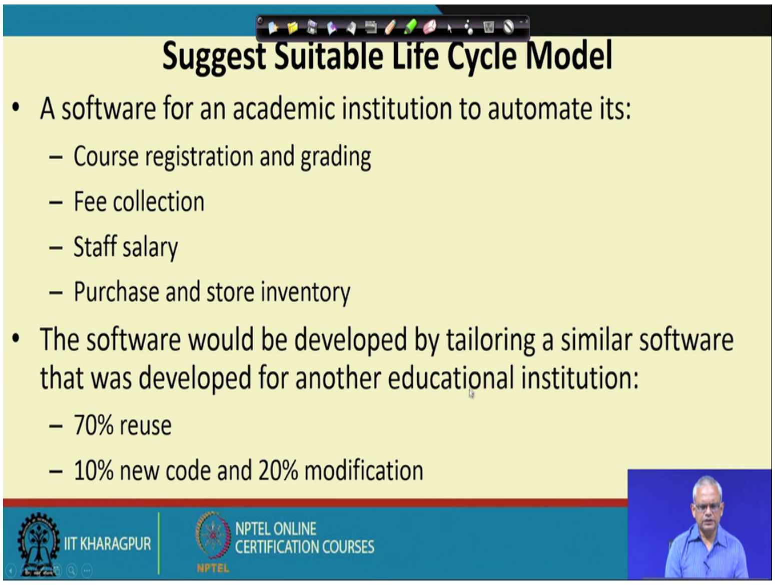
What are the documents that are produced at the end of every phase? Now, next question is what are the disadvantages of the iterative waterfall model? If, you recollect one major disadvantage is with respect to accommodation of change requests the customer is required to identify all the requirements upfront. The customer has to visualize what the software will be and it is extremely tough job it is a hard to identify all, the requirements that are required at the beginning and invariably 40 percent or 50 percent of the requirements changed during the development time itself.

And therefore, the iterative waterfall model is very inflexible. The second problem with the iterative waterfall model is that it is a heavy weight model emphasizes production of documents, rather than production of code in increments. The third disadvantage may be that it does not allow overlapping of the phases and so on. We have discussed several points and disadvantages of iterative waterfall model please review them why has agile model become so popular ok.

We discussed that the software projects have changed over here, the projects are now a short duration lot of code reuse been made and now the customer satisfaction is very important need to deploy incremental software at the customer site and so on. What difficulties might be faced if no lifecycle model is followed for a certain large project?

So, that is basically exploratory lifecycle model is followed ok. The answer would be that the project may not complete there will be too much of cost overrun, it will cost too much, it will take too much time to develop or it may not complete at all the quality of the software will be poor and so on.

(Refer Slide Time: 17:46)



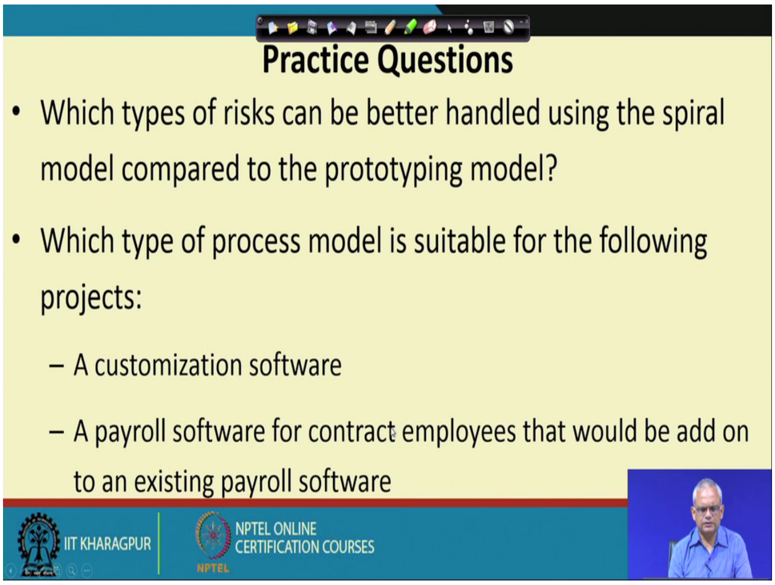
Suggest Suitable Life Cycle Model

- A software for an academic institution to automate its:
  - Course registration and grading
  - Fee collection
  - Staff salary
  - Purchase and store inventory
- The software would be developed by tailoring a similar software that was developed for another educational institution:
  - 70% reuse
  - 10% new code and 20% modification

Now, suggest a suitable lifecycle model for a software, which is to be developed for an academic institution to automate it is activities like course registration and grading fee collection, staff salary purchase and in store inventory. The software would be developed by tailoring a similar software that was developed by developed for another educational institution, it is expected that there will be 70 percent reuse of the code, 10 percent of new code to be written and 20 percent modification will be done.

Waterfall model is clearly unsuitable for this, because it will involve tailoring a similar software. So, an agile model can be suitable answer for this.

(Refer Slide Time: 18:43)



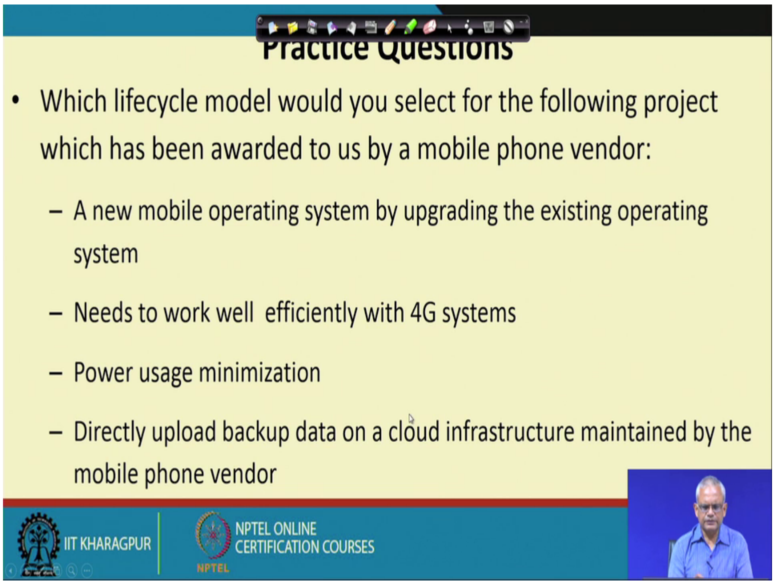
Practice Questions

- Which types of risks can be better handled using the spiral model compared to the prototyping model?
- Which type of process model is suitable for the following projects:
  - A customization software
  - A payroll software for contract employees that would be add on to an existing payroll software

Which types of risk can be better handled using the spiral model compared to prototyping model. In prototyping model the risks, which can be identified upfront those can be handled, but in the spiral model the risks which appear after the development starts, they can be handled better, which type of process model is suitable for the following project customization software ok. An agile model payroll software for contract employees that would be an add on to an existing payroll software.

So, there is basically a small increment to a existing software ok, this can be also an agile model.

(Refer Slide Time: 19:36)

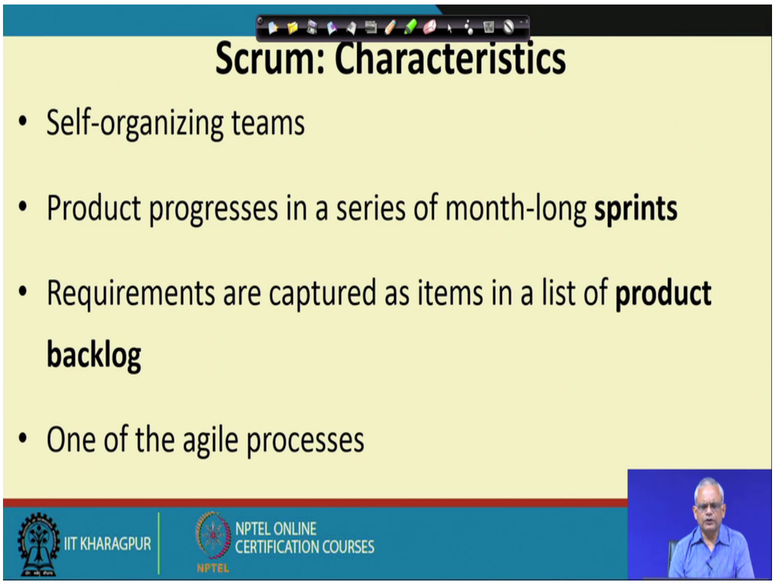


Practice Questions

- Which lifecycle model would you select for the following project which has been awarded to us by a mobile phone vendor:
  - A new mobile operating system by upgrading the existing operating system
  - Needs to work well efficiently with 4G systems
  - Power usage minimization
  - Directly upload backup data on a cloud infrastructure maintained by the mobile phone vendor

Which, lifecycle model would you select for the following project which has been avoided to us by mobile phone vendor? A new mobile operating system upgrading the existing operating system needs to work well efficiently with 4 G systems, needs to do power uses minimization, directly upload backup data on cloud infrastructure maintained by a mobile phone vendor. So, there is lot of challenge in this project new technology and therefore, agile model will be suitable. Now, let us look at one more agile model which is also very popular the scrum.

(Refer Slide Time: 20:20)



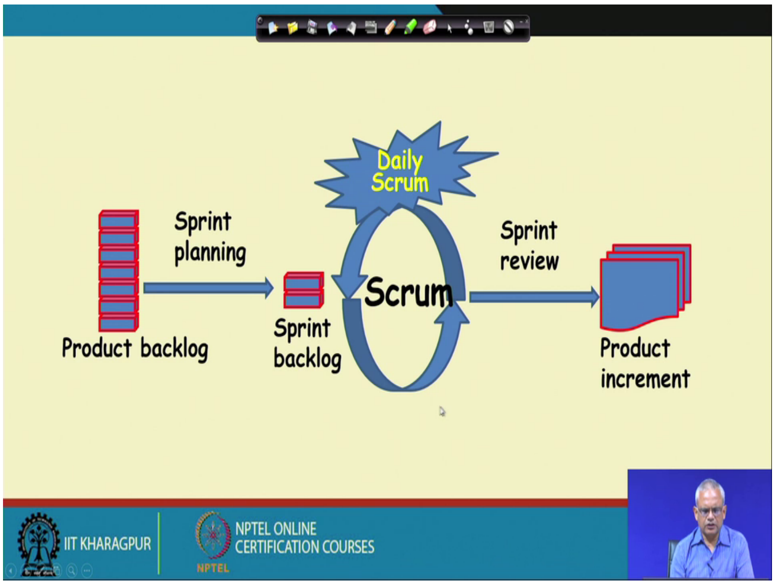
Scrum: Characteristics

- Self-organizing teams
- Product progresses in a series of month-long **sprints**
- Requirements are captured as items in a list of **product backlog**
- One of the agile processes

The scrum model has all the characteristics of an agile model, but it has it is own characteristics over the agile model; one is self-organizing teams, the self-organizing teams means that the small team they decide who will do which work.

Here, the terminologies are also bit different the increments are about a month and these increments are called as sprints. And, the requirements are captured as product backlog that is also another terminology with this is the software requirements are captured continuously, and those which are yet to be developed are called as product backlog, this is the crux of the scrum model.

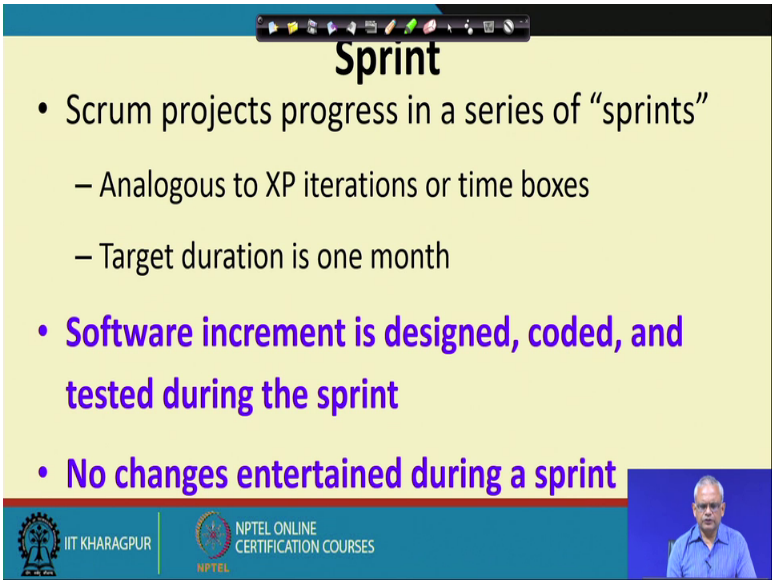
(Refer Slide Time: 21:26)



The product backlog keep on arising and in the sprint planning meeting one of the product backlog that is a user story these are all user stories. So, appear as product backlog some of them are chosen for the next sprint, a sprint is about a month long and this form the sprint backlog.

And, then every day the developers meet. So, this is the scrum every day the developers meet the daily scrum, they identify if any problem is being faced. And, then they provide any solution and then after the daily 10 15 minute meeting they go to do their work. And, the sprint backlog after it gets completed then there is a sprint review meeting. And then, the product increment is produced after the sprint review meeting is successful the product increment is deployed at the customer site.

(Refer Slide Time: 23:04)

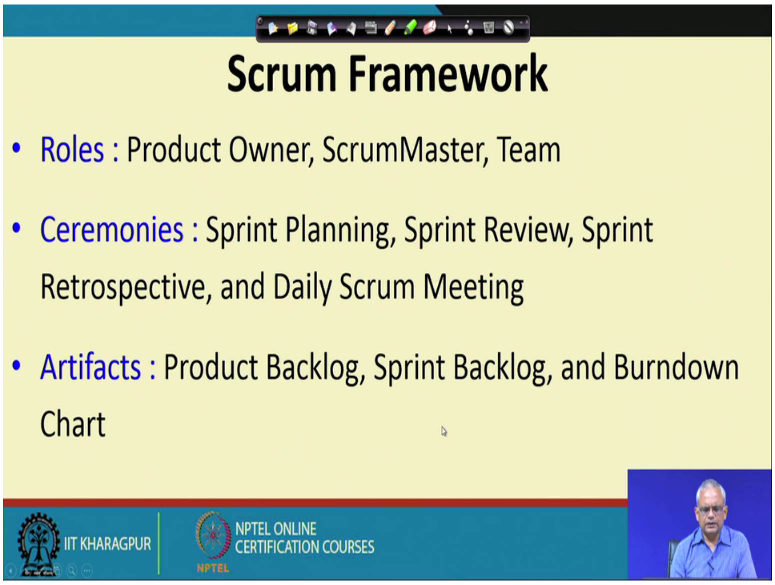


Sprint

- Scrum projects progress in a series of “sprints”
  - Analogous to XP iterations or time boxes
  - Target duration is one month
- Software increment is designed, coded, and tested during the sprint
- No changes entertained during a sprint

Here, the project progresses in a series of “sprints” these are similar to the iteration time box is the increments and typically about a month is the duration. In this sprint software increment is designed, coded, and tested, but one thing is that once the sprint backlog is formed during the sprint for that one month no change to that is entertained. The idea here is that, when the short term plan is made and the team is working on it we should not disturb it otherwise it will never converge.

(Refer Slide Time: 24:01)



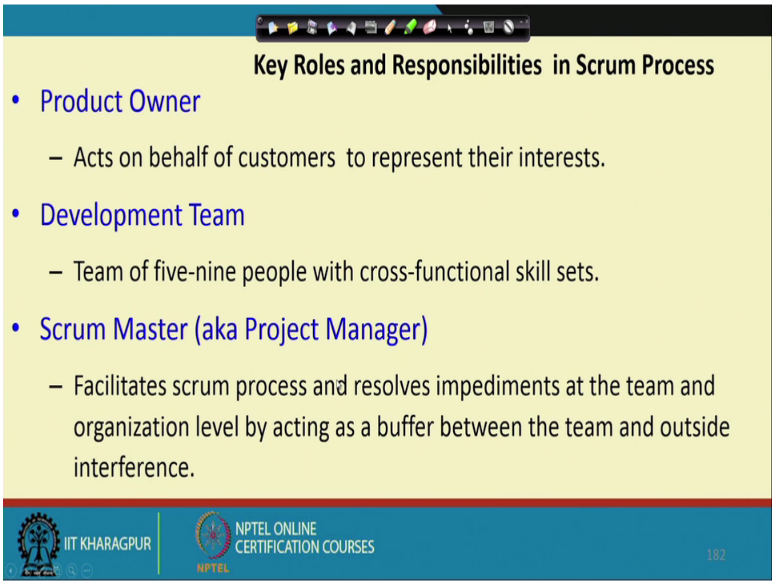
Scrum Framework

- Roles : Product Owner, ScrumMaster, Team
- Ceremonies : Sprint Planning, Sprint Review, Sprint Retrospective, and Daily Scrum Meeting
- Artifacts : Product Backlog, Sprint Backlog, and Burndown Chart

Here, there are some terminologies which constitute the scrum framework, there are 3 roles in the team one is the product owner. The product owner is either a customer representative or one of the development team member he acts as if he is the customer he is called as a product owner, the scrum master who is actually the project manager and then the team members. There are some meetings here these are called a ceremonies the sprint planning meeting, sprint review meeting, sprint retrospective meeting, and daily scrum meeting.

There are some documents which are used the product backlog, which is all the requirements the sprint backlog, which is basically the requirements per one sprint and then various types are Burndown charts, which capture how far the project has progressed.

(Refer Slide Time: 25:14)



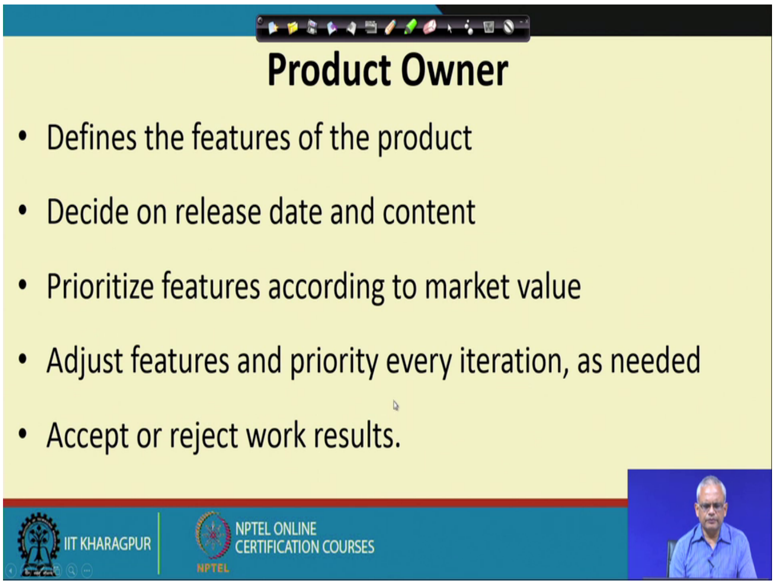
Key Roles and Responsibilities in Scrum Process

- Product Owner
  - Acts on behalf of customers to represent their interests.
- Development Team
  - Team of five-nine people with cross-functional skill sets.
- Scrum Master (aka Project Manager)
  - Facilitates scrum process and resolves impediments at the team and organization level by acting as a buffer between the team and outside interference.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

The product owner acts on behalf of the customer he may be a customer representative or maybe a team member who acts like the customer. The development team about 5 to 9 people with cross functional skills like testing GUI development database development and so on. The scrum master is the project manager; facilitates the scrum process and resolves any problems that the team may be fashion. And, also interacts with the customer and also the top management and shields the team from outside influence.

(Refer Slide Time: 26:02)

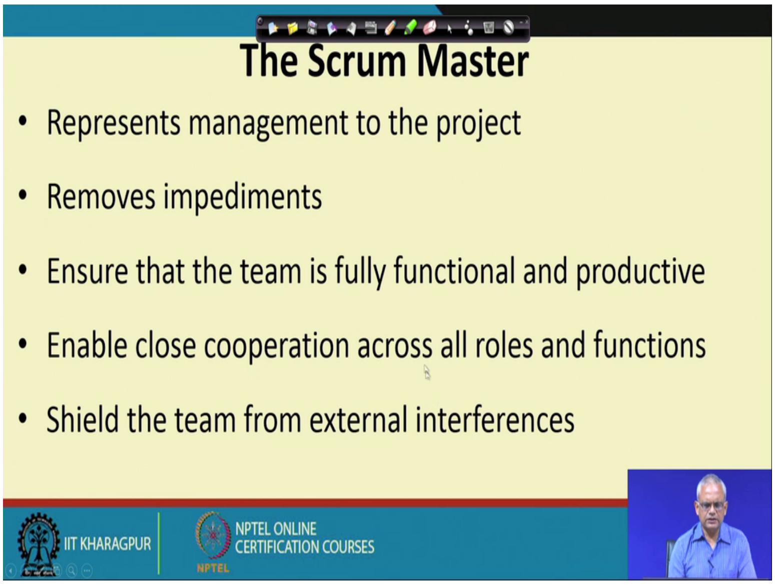


Product Owner

- Defines the features of the product
- Decide on release date and content
- Prioritize features according to market value
- Adjust features and priority every iteration, as needed
- Accept or reject work results.

The product manager the product owner acts as if the customer defines the features, decides on the release date, prioritizes the features according to what is his requirement? Adjust the features and priority value and then at the end of a sprint either accepts or rejects the result.

(Refer Slide Time: 26:30)

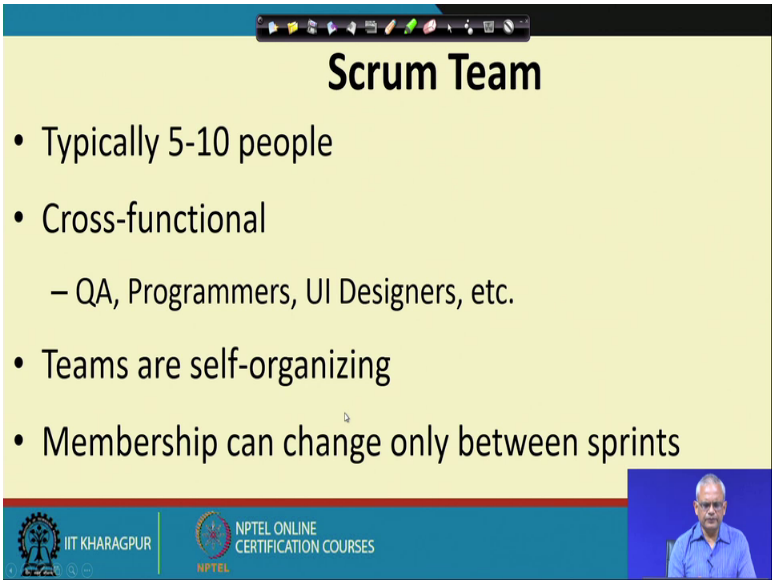


The Scrum Master

- Represents management to the project
- Removes impediments
- Ensure that the team is fully functional and productive
- Enable close cooperation across all roles and functions
- Shield the team from external interferences

The scrum master is actually the project manager, removes any problems like a infrastructure etcetera. Ensures that the team is fully functional enables cooperation and seals the team from external interfaces.

(Refer Slide Time: 26:53)



Scrum Team

- Typically 5-10 people
- Cross-functional
  - QA, Programmers, UI Designers, etc.
- Teams are self-organizing
- Membership can change only between sprints

The scrum team members are actually 5 to 10 people they are have expertise in various cross functional areas like quality assurance, coding, user interface design, database etcetera. The team members are self-organizing that decide who can do the work best and then they decide among themselves. And, in between one sprint membership is not changed new members can be added only at the end of a sprint.

We are towards the end of this lecture hour, we will stop here. And we will continue from this point in the next lecture.

Thank you.