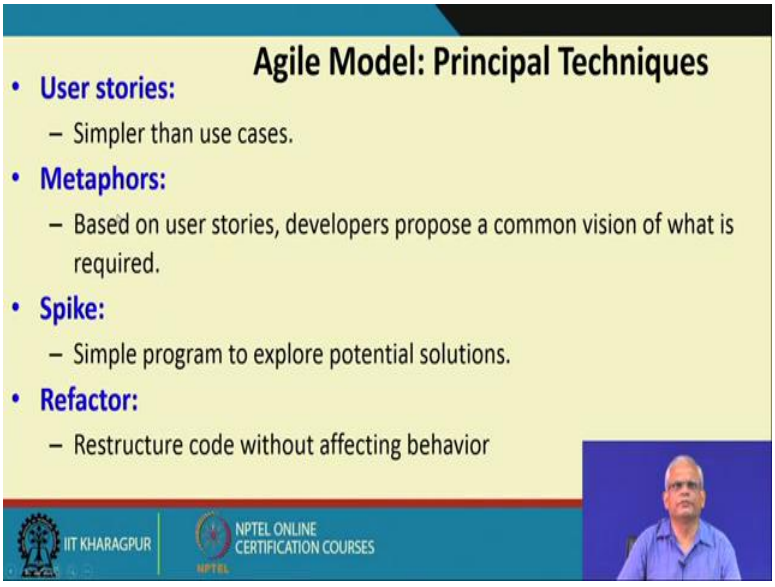
Software Project Management
Prof. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 09
Life Cycle Models - V

Welcome to this lecture. In the last lecture, we had discussed about the evolutionary model, the spiral model and just introduced the agile development. The agile models have become very popular. We will see that these have lot of advantages especially for the projects that are being undertaken now. At the beginning, we had said that the type of projects that are undertaken have undergone a drastic change over the years. Now the projects are of very short duration; 1 month, 2 month or 3 month projects are very common whereas, earlier there were multiyear projects; 3 year, 4 year, 5 year projects and now lot of reuse is being made only customization work.

So, that is service type of projects. Earlier we had product development from scratch, now it is customization. Slowly most of the projects are becoming like this that short development deployment at the customer side getting feedback and so on. And in that respect, the agile model is very advantageous and we had just started discussing about the agile model.

(Refer Slide Time: 01:45)

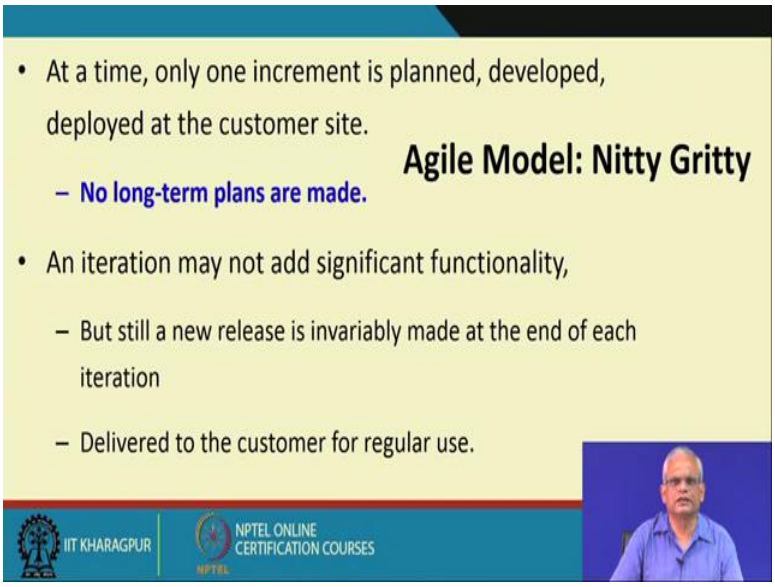


Let us look at more details of the agile model. Last time we had seen that the agile model incorporates some features like customer participation, incremental development, integration and testing over each iteration, deployment at the customer site, less documentation and so on.

Now, let us look at the main techniques of the agile model. Here no formal requirements are developed; it is based on user stories; as the name says that these are more informal than a requirements specification. These are like stories; these are simpler than the use cases. To give a overall design perspective, agile model proposes metaphors where there is a common vision of what is required and based on that the development starts. Wherever required a spike is done; a spike is a simple program that is written to explore potential solution. We can see that it is similar to a prototype. Wherever there is uncertainty develop a spike, check out the alternative whether the spike performs well and so on and also refactor.

Once it is developed at the customer site and the customer accepts it, restructure the code without affecting the behavior such that the code becomes more structured, put some design into the code. So, here as you can see that the design after each iteration is afterthought, the code is restructured. Initially the code is made to work and then it is restructured of course, we have the metaphor where there is some common vision overall design and so on.

(Refer Slide Time: 03:51)



It is incremental; each time one increment is planned developed and deployed at the customer site, no long term plans are made. Iteration may not add significant functionality, it may just only enhance the existing functionality.

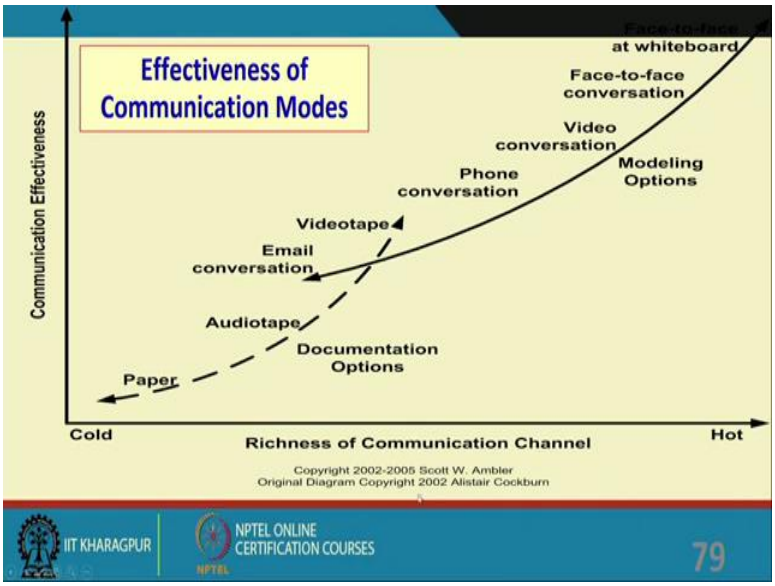
But then at the end of each iteration invariably, code is deployed at the customer site and the length of the iteration is usually fixed something like 2 to 4 weeks. And remember here that after each iteration, the customer makes the code; puts the code into regular use. It is not that they just evaluate it, they just they start using it regularly.

(Refer Slide Time: 04:50)

As we are saying that the agile model, one important thing is face to face communication. This required that the developers share the same building or the same room they meet regularly discuss issues rather than passing documents to each other they go and explain to each other what is required and so on. To facilitate face to face communication, they share a single office space and typically the team size is small 5 to 9 people so, that they can interact well.

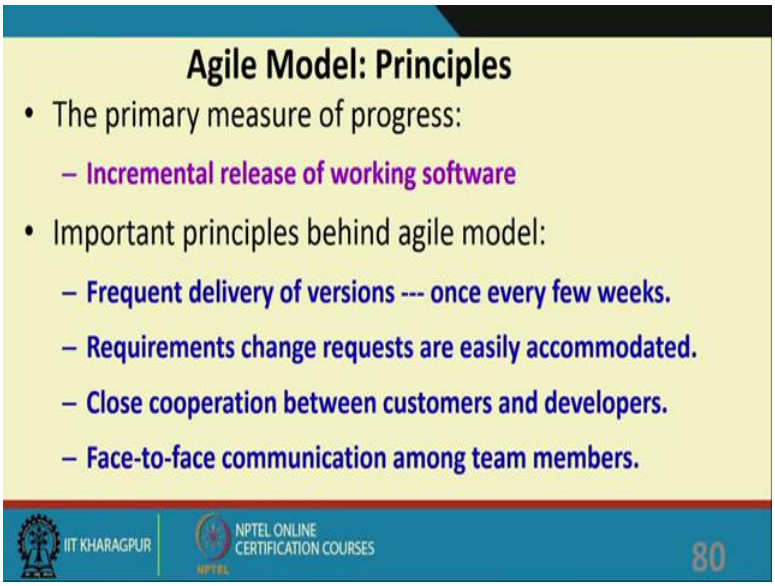
And this makes the agile model very suited to small projects, but of course, there have been effort to use this to scale this to larger projects. But then it started with focused for small projects. One thing need to mention here is that when want to convey something to somebody, it is not a good idea to give him a document to read. If you can explain him over a whiteboard that is the best way, you can explain him; you pass a document or explain him on phone or email that is not a good idea.

(Refer Slide Time: 06:16)



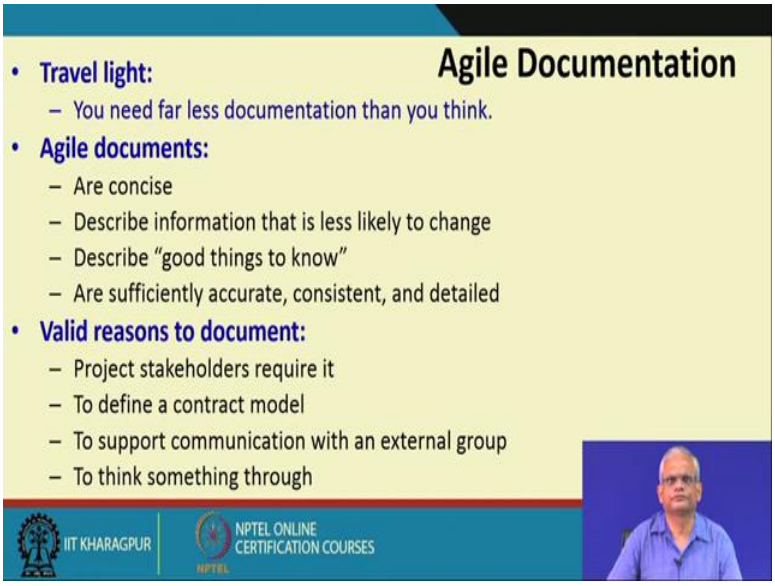
In fact, there are experiments done to find the effectiveness of various communication modes. This is given by Alistair Cockburn from experiments; he could determine that paper is a cold form of communication not very effective. If you want to convey somebody something give him a paper specially technical communication is not a good way to communicate. Audio tape is slightly better than paper. Email conversation may be slightly better; videotape slightly better, video conversation is still better and face to face conversation with a whiteboard. This is the most effective way to communicate technical items; this was the finding here. So, a small team they meet with a whiteboard and one person explains his ideas on the whiteboard that is the most effective way to have the communication and that is incorporated in this model; face to face communication is give given importance.

(Refer Slide Time: 07:040)



Here the progress is made measured in how many increments have been deployed at customer site. There is a frequent delivery of versions once every few weeks. The customer uses the delivered versions and gives requirement changes. These are accommodated is a close cooperation between the customer and developer and among the team members there is face to face communication. These are some of the very fundamental principles of the agile model.

(Refer Slide Time: 08:20)

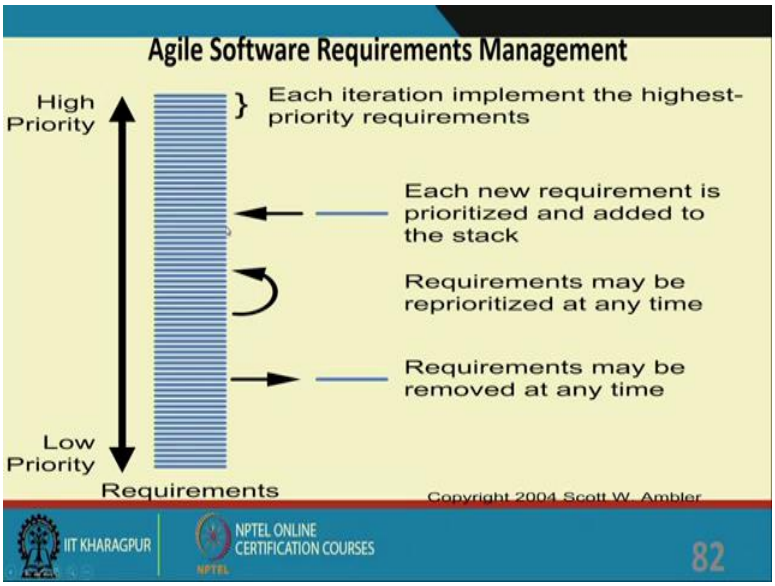


But what about documentation is no documentation done in agile model, not really. Here the idea is to travel light that is you do not document unless it is needed. You need far less documentation than you think. In the waterfall model, everything needs to be documented; the review results, the plans, the requirement review, the design, high level design, high level design review everything is documented. But here, the main idea here is that the documents need to be prepared only when these are required by somebody to be referred over a long time. But even then these are concise; these are the document is prepared only for information that is not likely to change that is at the end of the development things do not change.

So, the end of development documents are prepared and here only those things are documented where somebody can learn something which will be very informative to somebody which is good things to know. It is not that for the sake of documentation, documentation is created. The documents are sufficiently accurate, consistent and detailed because these are prepared when there are no more changes. Some of the valid reasons to document are that the project stakeholders need it; for example, the customer needs it maybe there is a contract form to be signed for which it is needed, maybe there is an external group who like to review or give their suggestions, maybe the document will be for them, maybe the document will be for those to be referred after the project is over and so on.

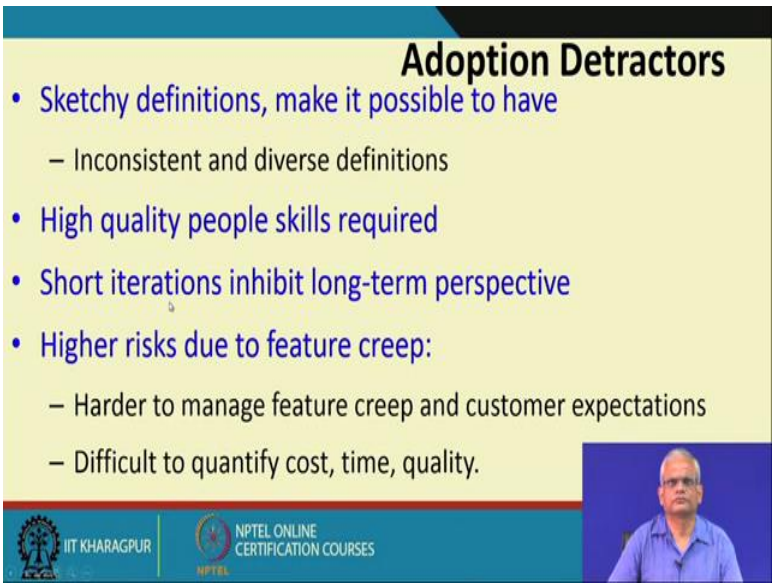
Only when there are valid reasons to document, the documents are prepared. It is not that documents are prepared by default as is the case with waterfall model.

(Refer Slide Time: 10:54)



The requirements here are changing over the development time and new requirements as they arise, they are kept in the requirements stack here. The stack is prioritized the top are the ones that will be first taken up. These are high priority and these are all requirements and this diagram given by Scott Amblers book the requirements keep on arising. They are inserted in the stack, they are taken out removed whenever necessary they change; they are reprioritized, but then there is a priority assigned to each requirement and then this are changed dynamically.

(Refer Slide Time: 11:47)

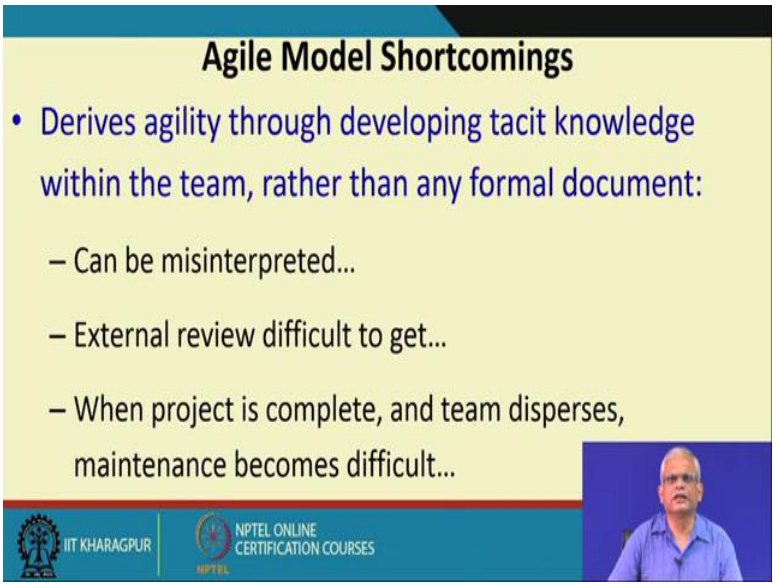


The agile model has many advantages, but then to make it practically useful, you must be aware of some pitfalls. First thing is that here documents are not there. It is based on explanation on a whiteboard and so on. And therefore, somebody can misunderstand.

High quality people skills are required who when in doubt will consult each other and do what is correct. Long term design is not made and the short iterations, these degrade the design structure. There can be feature creep feature creep is the word where the customers or the developers become more ambitious. They just keep on thinking new features which can be incorporated without thinking of whether how much the feature will be used; what will be the value to the customer, what will be the cost of developing it, feature after feature get added. So, there is a higher risk of feature creep in the agile model because the freedom is given to give feedback and request for new features and so on.

And therefore, it becomes difficult for the project manager to manage the feature creep and also since the development is dynamic, the features change new features get added and so on. To give an upfront cost to a development becomes very very difficult; to give a upfront timeline by which development will be done is difficult and quality difficult to assure sure much more difficult.

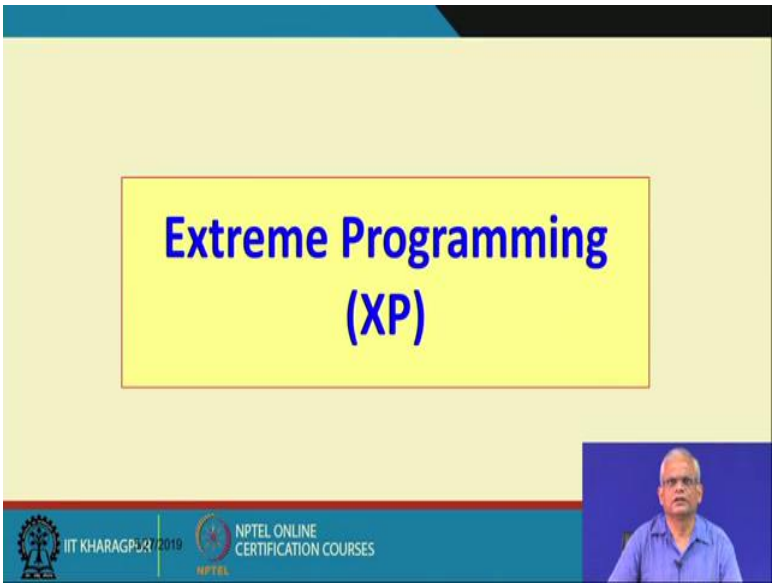
(Refer Slide Time: 13:56)



The document a done away that is a good thing saves time effort of preparing document, but then must be aware that the lack of document. There can chances of

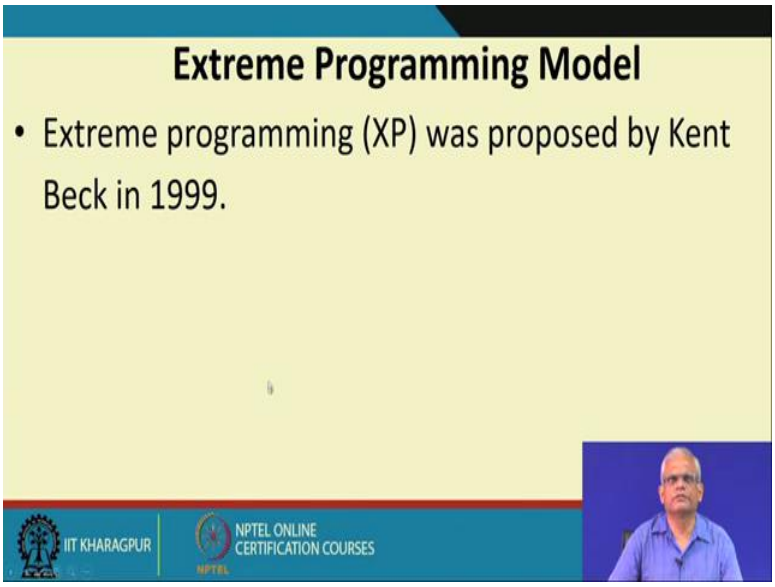
misinterpretation, getting reviews on the document are difficult and also when the project is complete unless the documents are consciously prepared at the end of the project before the team disperses, then maintenance may become difficult.

(Refer Slide Time: 14:30)



With this overall understanding of the agile model, let us look at some specific agile processes. One of the agile process which is very popular is the Extreme Programming which is also known as the XP model.

(Refer Slide Time: 14:48)



It was proposed by Kent Beck, 1999.

(Refer Slide Time: 14:54)

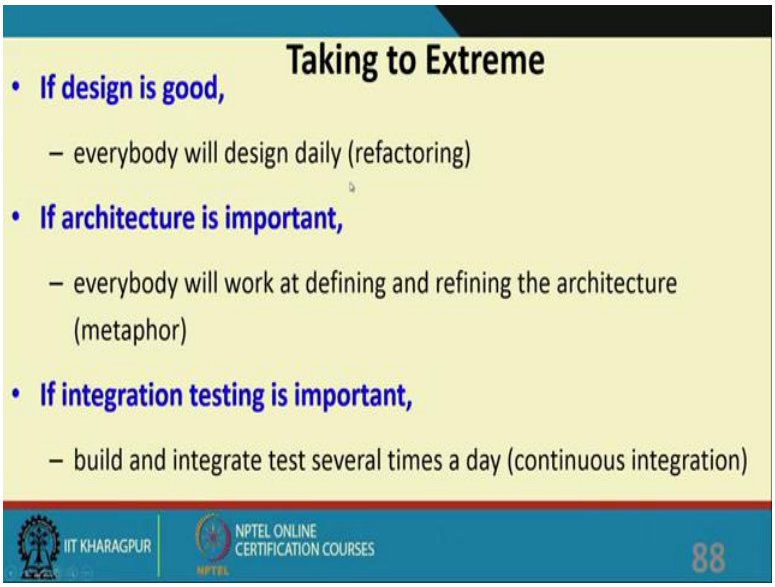
The main items that are introduced here is pair programming. In a pair programming, each desk each desktop is manned by two programmers. The main idea here is that reviewing is a good thing; so, why not continually review each other's work. When one programmer writes the program, the other programmers goes through the code and reviews it give suggestions how to make it better code, more efficient code, avoid mistakes and so on and the take turn one programmer program programs for an hour maybe writes few functions and then the other programmer takes up, write few more functions where as the first programmer reviews that.

Here every day the testing takes places unlike the waterfall model where testing is at the end. Here the test cases are written continually and the test cases are executed before a feature is passed. Before a feature is implemented, the test cases are written for that feature and the feature is considered passed when it passes the test cases. This is called as test driven development.

Here incremental development is practiced every few day increments are given. The name extreme programming comes from the fact that the good practices are taken to extreme. The good practices are code review is good and this is implemented in the form of pair programming testing is good; good practice and therefore test driven development. Incremental development is good therefore, every few days new increments are develop in delivered. Simplicity is good and therefore, the simplest

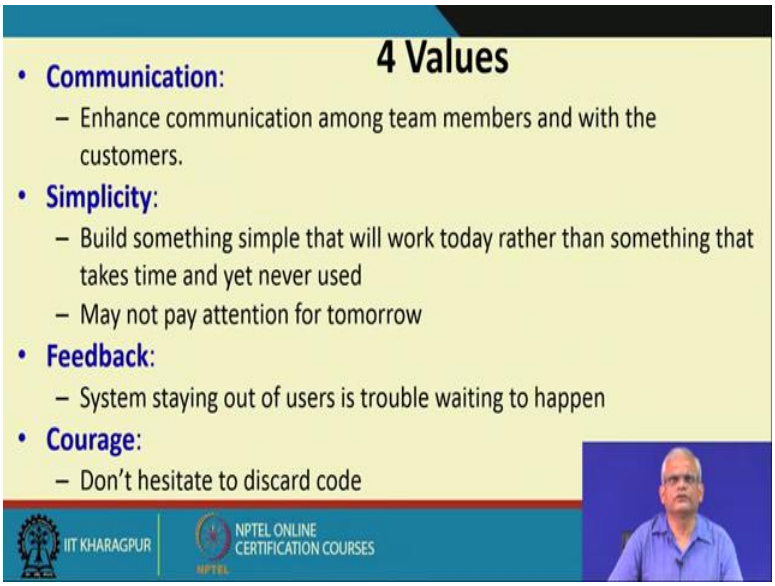
design is developed. Do not try to make it think of extensions that may be required decades later or long time, afterwards some enhancement may be required; do not think of it.

(Refer Slide Time: 17:45)



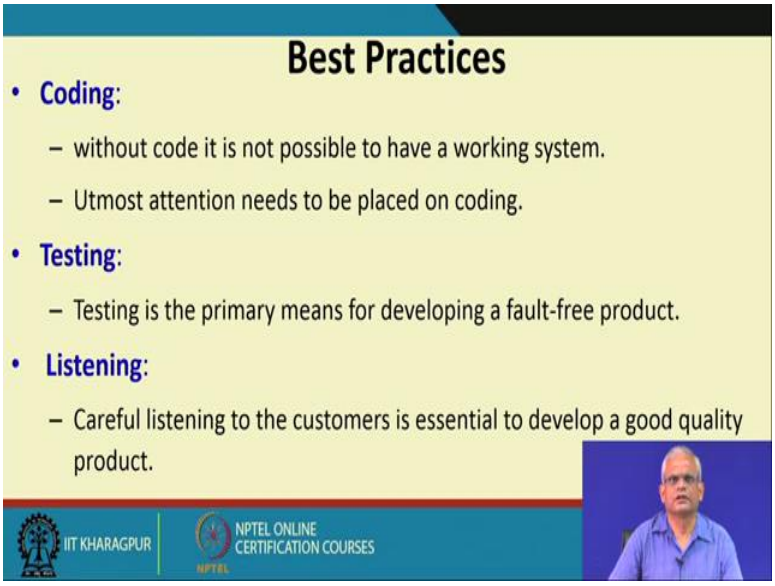
Now, make it work make the simplest design make it work. Designing is good and therefore, after the software is deployed accepted, refactor the code, put some design, make the code better architecture is important and therefore, a metaphor has to be defined. Integration testing is important; therefore, several times a day build and integrate that is continuous integration as the development proceeds. Remember that this was one of the major problem with waterfall model where the integration took place only at the end.

(Refer Slide Time: 18:28)



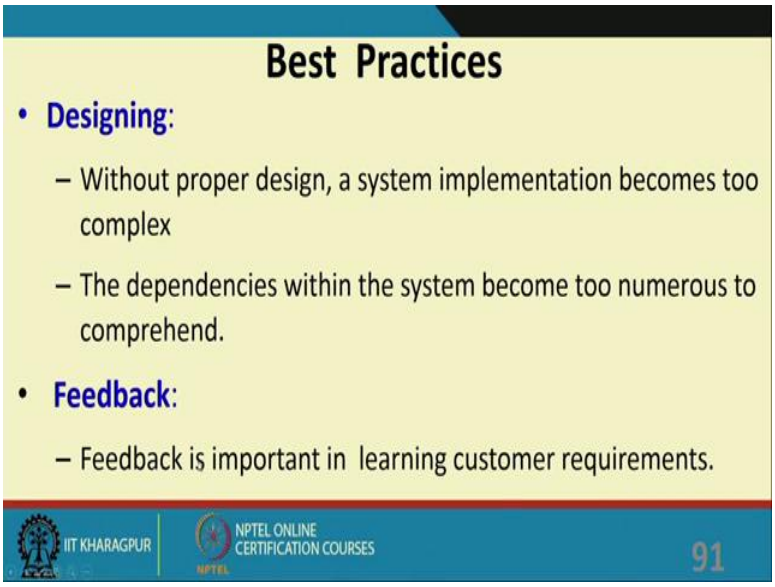
The four values here; communication face to face communication, simplicity, implement the simplest design may not pay attention for tomorrow make it work. Feedback get customer feedback, encourage customer feedback. If you do not get feedback from the customer, this is trouble going to happen because at the end they will reject; they will be unhappy. And here the other value is courage. If some code is not good discard it, rewrite it. Design is not good, discard the design; redo the design.

(Refer Slide Time: 19:15)



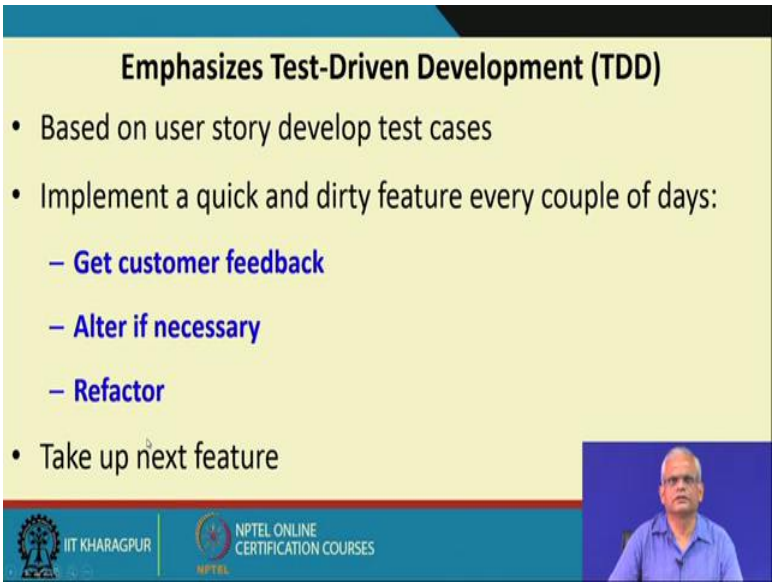
Coding is a best practice utmost attention on coding testing is a best practice primary means of developing fault free software and therefore, importance to testing and listening is a best practice listen to the customer and find out what is required really by the customer.

(Refer Slide Time: 19:40)



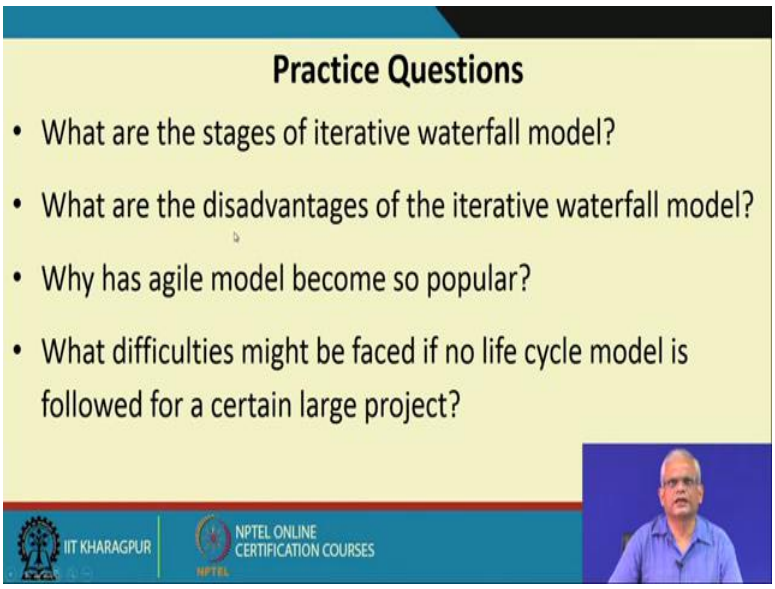
Designing without proper design, the system becomes complex. Therefore, put design into the system. And feedback is a important thing get customer feedback.

(Refer Slide Time: 19:59)



Emphasizes test driven development; based on the user story, first develop the test cases before writing the code first write the test cases that is the test driven development. First write the test cases based on the user story and then implement it. Once implemented run the test cases and the development is not complete until it passes all the test cases. Develop over increment every few days increments to be given, get customer feedback, alter if necessary and then finally, once accepted by the customer re factor, make the code better, put some design and then take up the next feature.

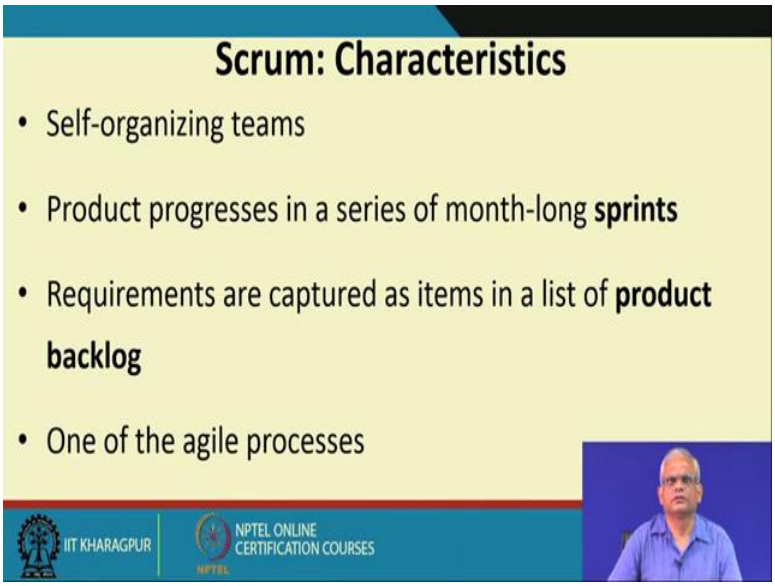
(Refer Slide Time: 20:56)



Now, let us look at few practice questions. So far, the points that have discussed; what are the stages of iterative waterfall model. Hope you remember, if not please look up; what are the major disadvantages of the iterative waterfall model.

Because the later models were proposed to overcome the disadvantages of iterative water fall model which you had already discussed. Several problems with the waterfall model, most important is difficulty in handling change requirements, why has agile model become so popular, what problems of the waterfall model it overcome, how it fits into the current type of projects, what difficulty might be faced if no life cycle model is followed in a certain large project ok.

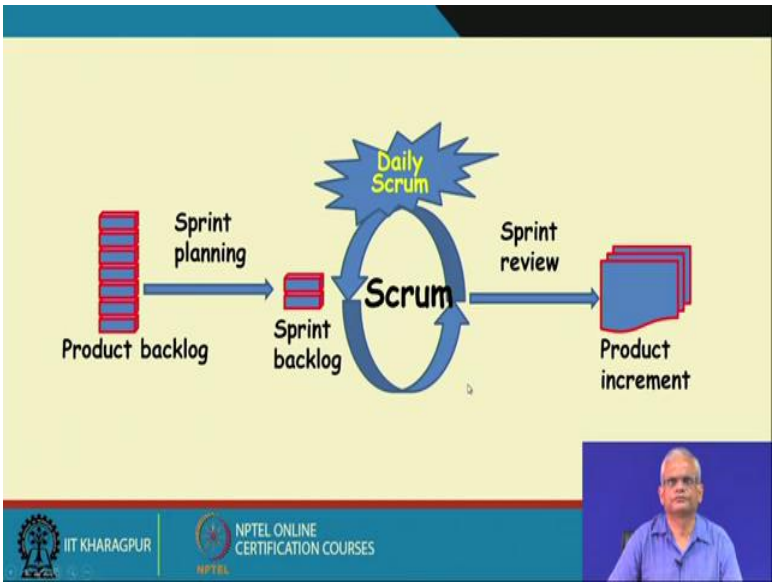
(Refer Slide Time: 22:09)



Now, let us look at another agile development process which is called a scrum. The main characteristics of scrum is self organizing teams that is the team members decide among themselves who will do what, who will do testing, who will do which function development etcetera. Here the product progresses in a series of month long sprints; the sprints are basically increments.

So, the increments are called here as sprints typically one month. Here the requirements are captured, we are looking at the requirements stack something like that which is called as a product backlog.

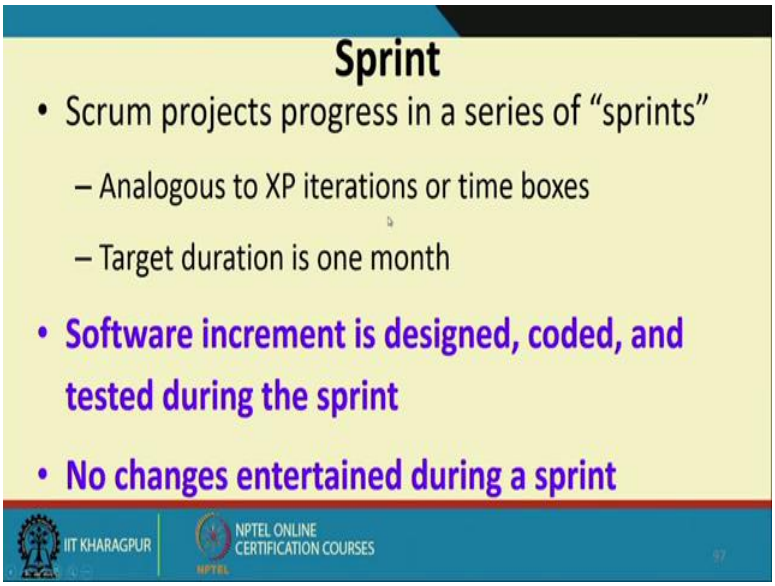
(Refer Slide Time: 23:02)



So, one of the agile processes; here is the requirements that have been gathered which arise as the development proceed some may get deleted, changed and so on. This is called as a product backlog. In each sprint, a sprint is a month long activity. One of the top priority feature requirement is taken out that forms the sprint backlog and once the sprint backlog is obtained, this is not changed anymore. It is immune to change and this is developed over a month long iteration here and every day the developers meet for a daily scrum meeting to review what has happened, what is the next thing to do and so on and they complete the sprint backlog.

In the sprint backlog, based on the identified requirements from here activities that need to be done to meet the requirements are identified and that are put in the sprint backlog which is put by the development team. And these activities are get completed over daily scrum and finally, the sprint review; the sprint is reviewed and the product increment is deployed at the customer site.

(Refer Slide Time: 24:32)

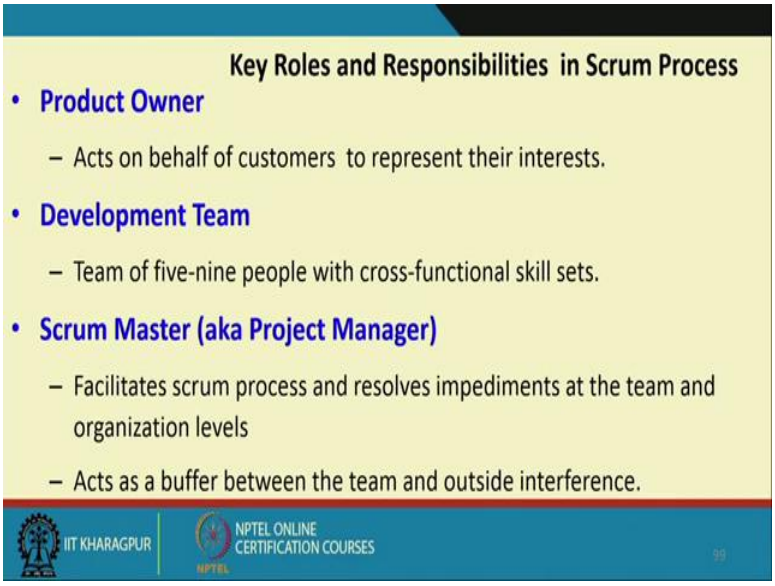


So, that is the main idea here. The progress is in the form of sprints, duration is typically one month. In one month, the some of the features from the product backlog are taken up; design, coded, tested. And once the sprint starts the requirements that have been taken up are not allowed to change otherwise the sprint will not converge. One of the principle here is that once the feature have been taken out from the product backlog, they are not allowed to change.

(Refer Slide Time: 25:11)

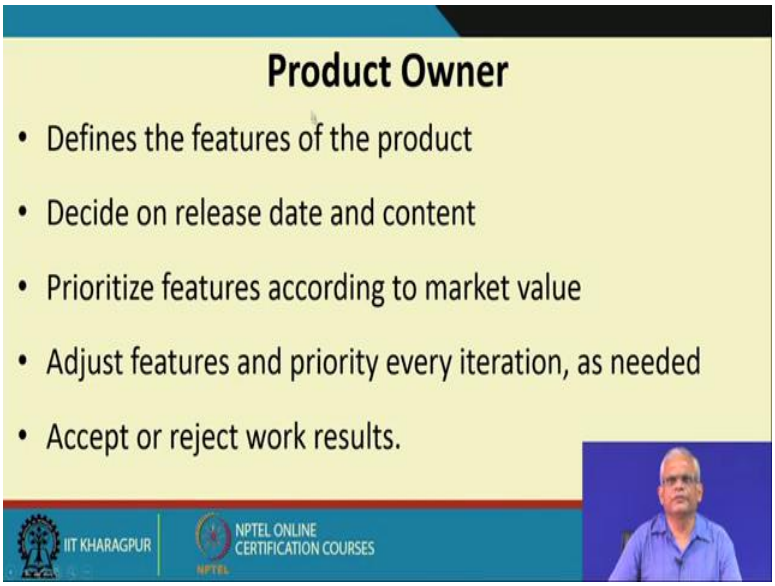
Here some of the terminologies the roles, ceremonies and artifacts. The roles are product owner. One of the team member acts as on behalf of the product owner; that is a customer. He has the customer perspective, the scrum master who is like a project manager and then the team members. There are various ceremonies that get conducted during development; one is the sprint planning, the sprint review at the end of a sprint, sprint retrospective and daily scrum meeting. There are various artifacts produced one is the product backlog which keeps track of the requirements that have been identified so far, more requirements can be identified as the development proceeds and these are kept in a prioritized order. The sprint backlog this is the activities to be done during the sprint. The burn down charts here how much progress has been made are depicted in the form of burn down charts.

(Refer Slide Time: 26:24)



The product owner has the customer perspective is the development team has five to nine people with cross functional skills. The scrum master is also called as a project manager. He facilitate the scrum is the buffer between the team and outside in interference and resolves any difficulties that that team members might be facing.

(Refer Slide Time: 26:57)



The product owner is one of the team members or maybe one of the customer representative who is part of the team. He acts as if the representative of the customer. He is the one who defines what are the features that will be required decides on the release date. Prioritizes features based on the customer perspective according to what will be the market value, adjust features and priority in the product backlog in every iteration and finally, accept or reject the results.

We have seen the main idea behind the scrum. There are many terminologies in the scrum; looked at some of the terminologies and we are just coming to the end of this lecture few more terminologies and concepts in the agile scrum model is could not discuss in this lecture. We will take up in the next lecture and will complete our discussion on the development life cycles in the next lecture.

Thank you.