Welcome to this lecture. In the last lecture had seen that evolutionary model and the incremental model, these are two very conceptually important lifecycle models. And, many successful models are based on these 2 principles. Looked at the RAD model and also the unified process, the unified process you said that it is extensively used for object oriented software development. It is a incremental model in the sense that much of the requirement is identified up front, again it has features of evolutionary because the features continue to be discovered as the development proceeds.
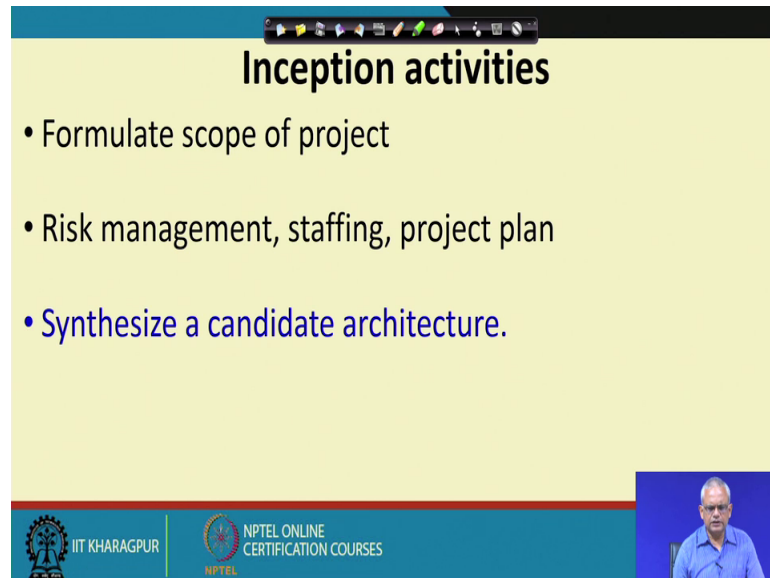
The development occurs over 4 phases.

(Refer Slide Time: 01:19)
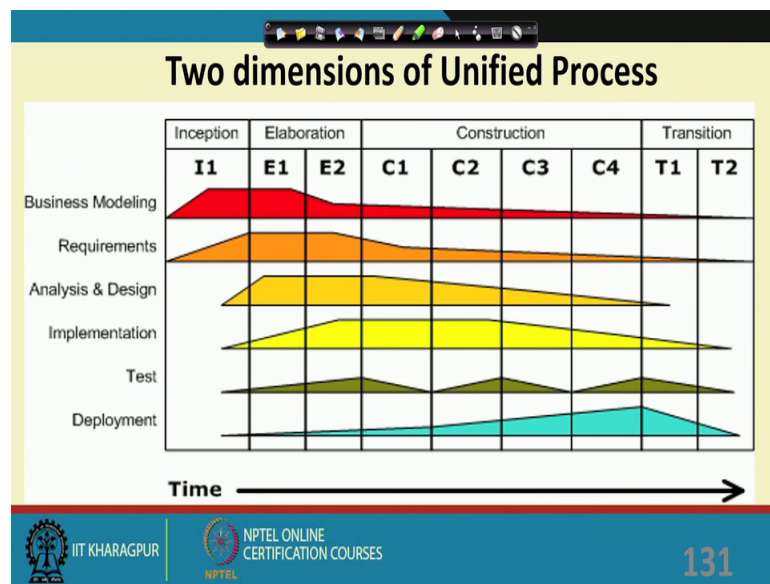
The inception, elaboration, construction and transition, and each phase can consists of several iterations. And, the life cycle activities the overlap across different phases, but then they peak during certain phases. For example, the business modeling that is study of how the business operates, peaks during the inception and elaboration. The requirement identification peaks during inception and elaboration and slowly tapers off.

Analysis and design start during elaboration and slowly start to taper off. Implementation peaks during construction and to some extent during the elaboration and then tapers off testing occurs all through. And then these are deployed as the increments result in deployable software these are deployed at the customer site, but the deployment peaks during the transition.
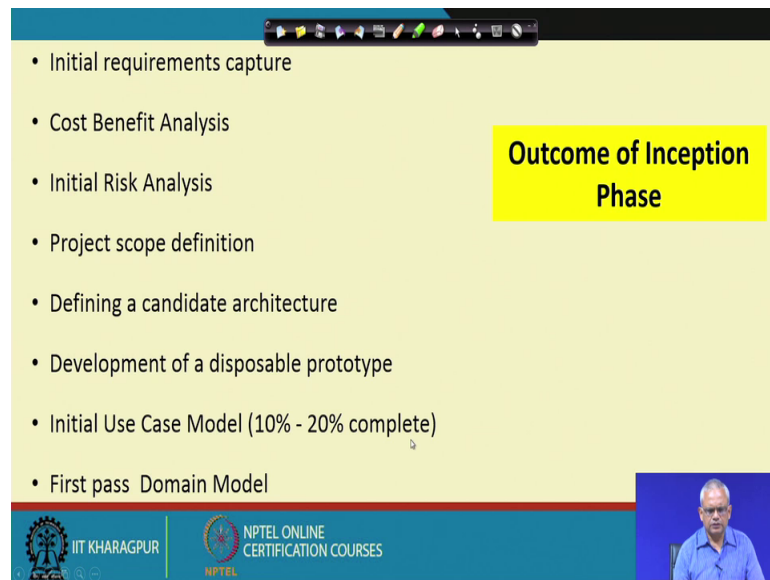
(Refer Slide Time: 02:50)

During the inception, the scope of the project is determined that is the features some plan documents like risk management staffing project plan are made. And, also an overall architecture is determined based on the project requirements.
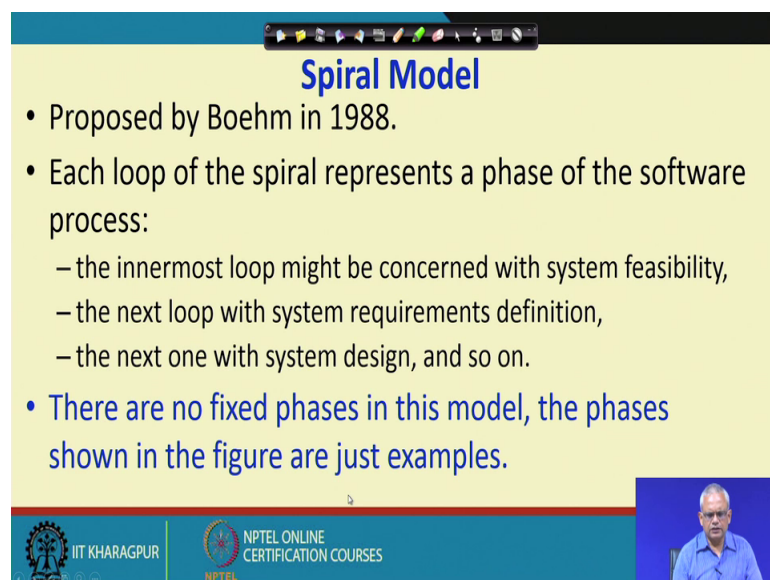
(Refer Slide Time: 03:22)

We will not spend more time on the unified process, we will just quickly look at the outcome of the inception page, initial requirements capture cost benefit analysis, risk analysis project scope definition, defining the candidate architecture, development of the prototype. It has disposable prototype development here to study between different architecture and the initial use case model and the first pass domain model.

(Refer Slide Time: 03:54)

Now, let us look at the spiral model which is a rather old model compared to the models that we just saw. The unified process, but it has been proposed by Boehm in 1988, it has

features of incremental development. And, also evolutionary development, here it is developed over several loops these are something similar to increments. The innermost loop may be concerned with system feasibility, the next loop with requirement definition next one with system design and so on. So, one thing that we can say is that in the spiral model each loop may not result in a deliverable software whereas, in the incremental model every increment actually leads to a deployable increment at the customer site.

The number of phases each loop is called as a phase here and there are no fixed phases in this model, the phases that we are drawn, in the next slide are just example, the number of phases are determined by the project manager and the team members as the development proceeds.

(Refer Slide Time: 05:37)

The phases are decided by the team members and each phase some risk is identified. And, these are resolved using prototype. So, one of the very important issue with the spiral model is risk handling. In every phase the most important risk that is being faced by the project is identified.

And desire resolved by developing a prototype; just contrast this with the prototyping model where only one prototype is made before the start of the project. So, the risks that could be identified the beginning of the project can be resolved in the prototyping model whereas, in the spiral model as the project continues more and more risks are identified and these are resolved.

Each loop is actually split into 4 sectors.

(Refer Slide Time: 06:56)
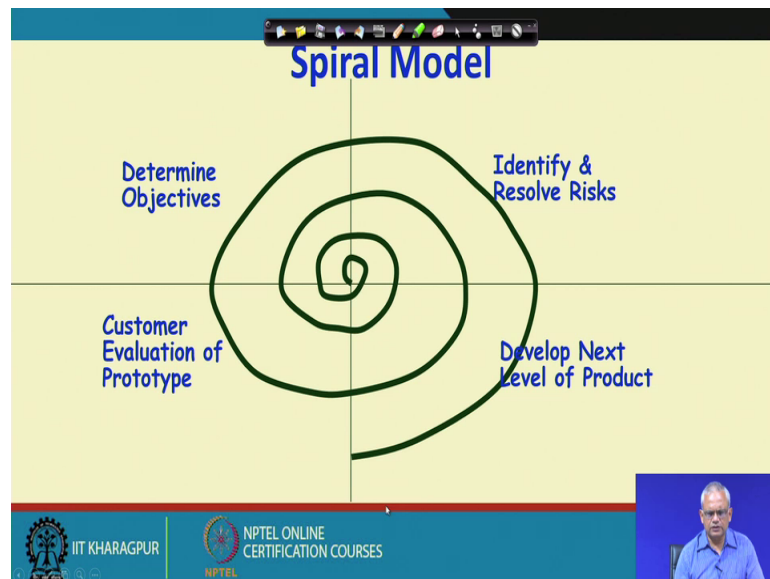
So, this is the spiral model and there are 4 quadrants here, each loop is called as a phase over each loop some risk is identified. And, then it is resolved through developing prototype and once the risk is resolved.

(Refer Slide Time: 07:22)
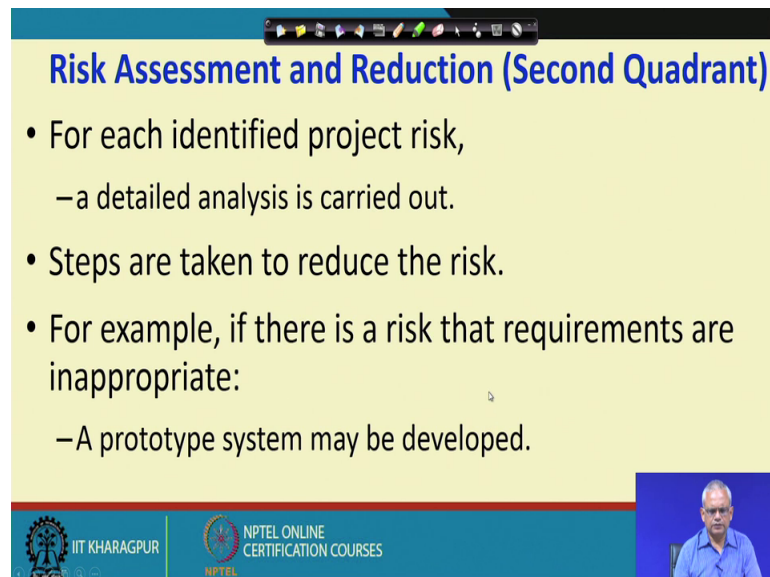
Then, the development occurs and finally, the customer site the customer evaluation of the prototype, there is mentioning that one of the important characteristic of the spiral model is risk handling. Risk is basically any adverse circumstance that might hamper the

successful completion of the software project. For example, whether the throughput will be sufficient to meet the customer requirement; in this case the prototype is built for resolving the issue and alternate solutions may be tried out and the best one may be used.
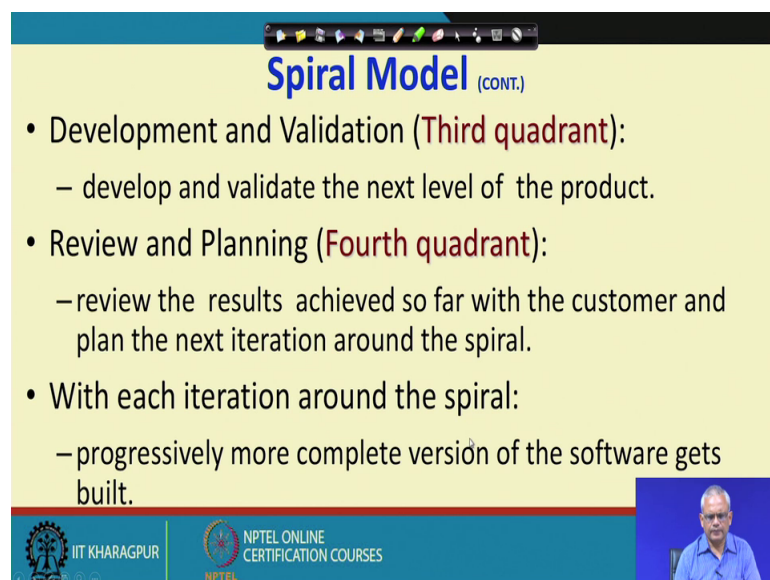
(Refer Slide Time: 08:25)

In the second quadrant detailed analysis of the identified feature is carried out, and then the risk that is identified is built through is a result through building a prototype.
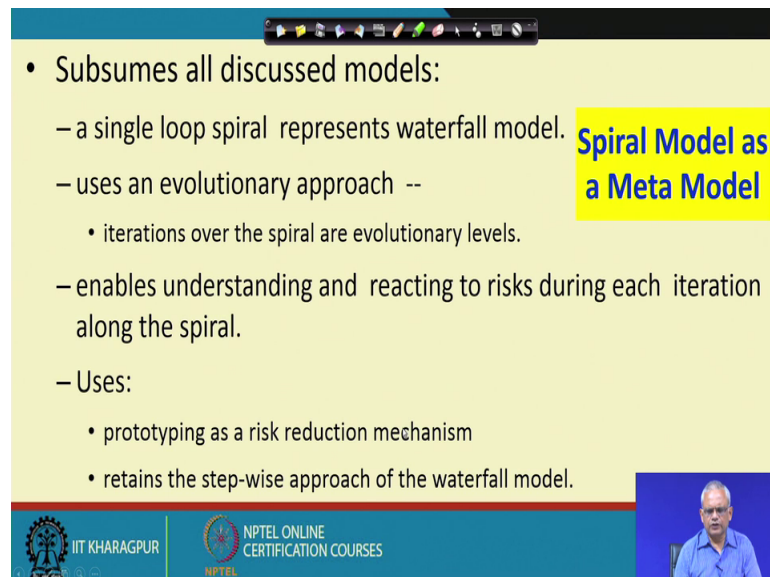
(Refer Slide Time: 08:45)

In the third quadrant, after the risk is resolved the development occurs and in the fourth quadrant review and planning. So, customer feedback is obtained based on the way the

risk was handled and development was done. And, with each iteration around the spiral more and more complete version of the software gets built, but then just remember that every spiral that is every phase may not lead to a deployable software at the client site.
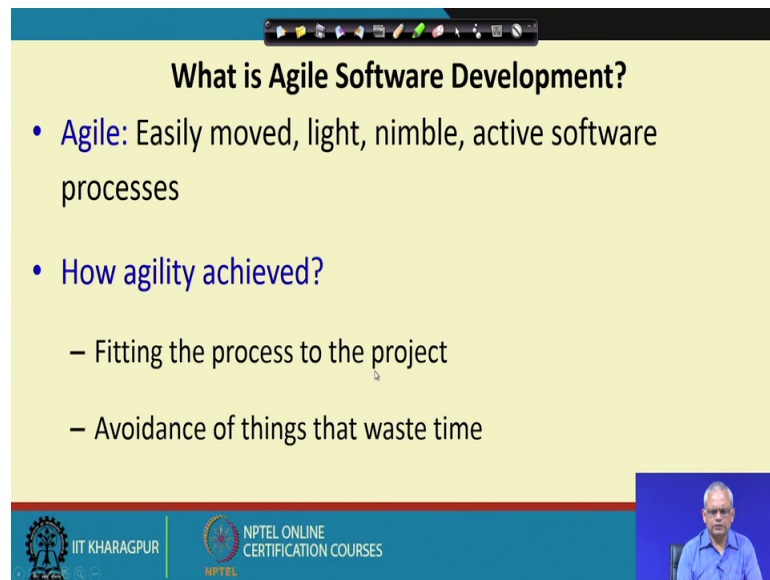
(Refer Slide Time: 09:21)

A spiral model is called as a Meta model, because it has features of the waterfall model, incremental model, evolutionary model, and so on. We can see that if we have a single loop of the spiral, then it is actually a waterfall model. And, then it has the features of the prototyping model, it has the features of the incremental and the evolutionary models. Now, let us look at the agile development models which have come into the picture for last 2 decades or.
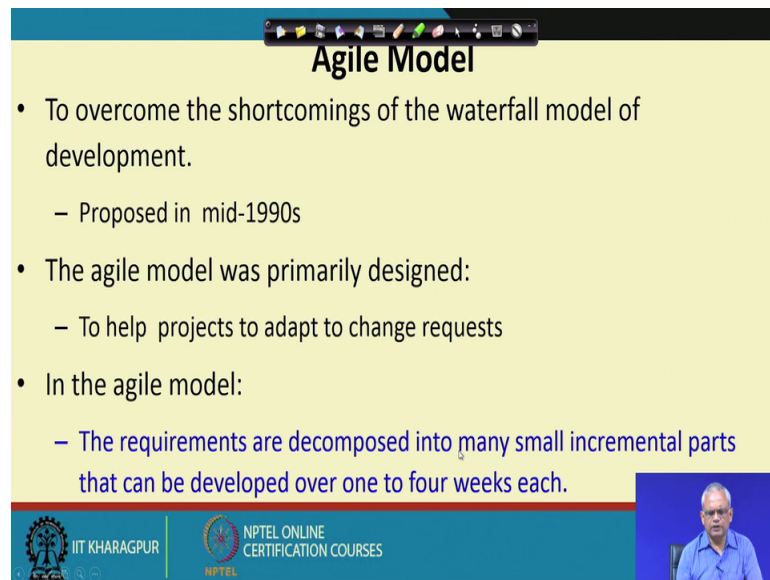
(Refer Slide Time: 10:07)

So, if you look at the meaning of agile in the dictionary, says fast development, light, weight, nimble is fast, active software process, but then how fast development is achieved. Here the agile model as you will see the details we will see that anything that wastes time are avoided, and also any activities that are not required are eliminated.

So, that we say that fitting process to the project. So, for a specific project any activities that are not required are eliminated. And, also any things that waste time is eliminated, but what things waste time. In the waterfall model who had seen that about 50 percent of the effort is spent on developing the documentation. And, some of this documentation are rarely used by anybody.

(Refer Slide Time: 11:38)

So, here among other things one thing is that it produces very little documentation. One of the major focus here is to facilitate the change requests from the customer and incorporate them efficiently. And, here it has features of the incremental and evolutionary model, just like the incremental model here these are the software is developed over increments and deployed at the client site. The time duration for each increment is about 1 to 4 weeks and this has the feature of the evolutionary model, because the requirements evolve over time.
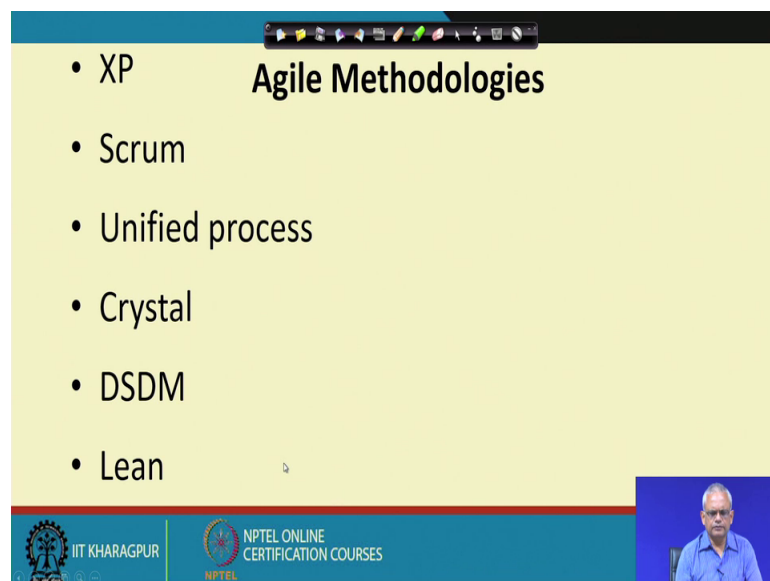
(Refer Slide Time: 12:35)

The Agile Manifesto was released towards the year 2000 and the Agile Manifesto is available at this website www dot agilemanifesto dot org. If, you look at the Agile Manifesto, it says that there are certain things that are important for the agile development. One is individual interactions are very important, these are much more important than the process itself or any tools that are used, working software over comprehensive documentation.

So, one of the things that is recognized here is that producing working code is much more important, then writing extensive documentation. Customer collaboration over contract negotiation, consciously the customer has to be involved in the project possibly by making some customer representatives part of the team. And, contract negotiation that is binding the customer to a contract. And then finally, making him sign that and obey that that is not the focus here and changes are welcome over following a plan.
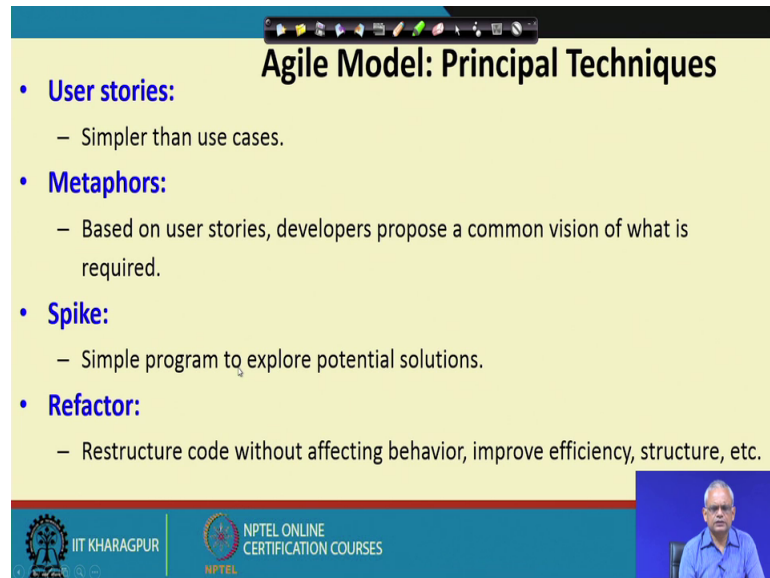
(Refer Slide Time: 14:16)

The agile methodologies are the agile is actually an umbrella term, many development methodologies actually qualify as agile, these are the features of the Agile Manifesto, extreme programming or XP, scrum, even the unified process, crystal, DSDM, lean etcetera.

(Refer Slide Time: 14:47)
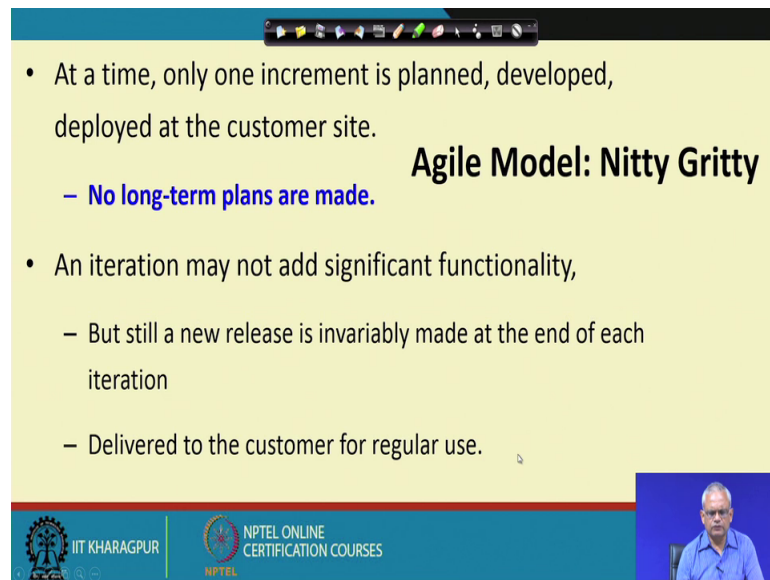
The agile model other than the 4 important manifesto, that we said it emphasizes many techniques. One is that the requirement should be in the form of a user, user story as the name implies it is bit informal.

So, understand the requirement informally, this is these are simpler than use cases and the development can start based in the stories. And, then based on the customer feedback this can be refined, it uses metaphors; metaphors are actually an overall design of the software is the common vision or the overall design of the software. Based on the various user stories requirements a overall design is obtained which is the metaphor.

A spike is like a prototype it is a program written to explore the potential solutions and evaluate the alternatives. And, once the software works and the client agrees to that, after that design is put into it. It is refined made structured and some design is put into it without affecting the behavior of the code, it may improve efficiency design structure etcetera.

(Refer Slide Time: 16:48)

This incremental model at a time only one increment is planned focus is on one increment at a time and that is deployed at the customer site. No long-term plans are made, each iteration may add even a small amount of code minor increments, but still after a time box an increment is made and that is deployed at the customer site. Sometimes the incremental feature may be very small, but still these are deployed. These are for regular use of the customer.

(Refer Slide Time: 17:43)

One of the very important aspect of the agile methodology is face to face communication, written document are not in favor the team members should communicate to each other rather than passing documents to each other. And, to facilitate this, the development team shares the same office space and the team size when it is small it works the best 5 to 9 people, and if the team size is so, small then it is suited for projects that are small.
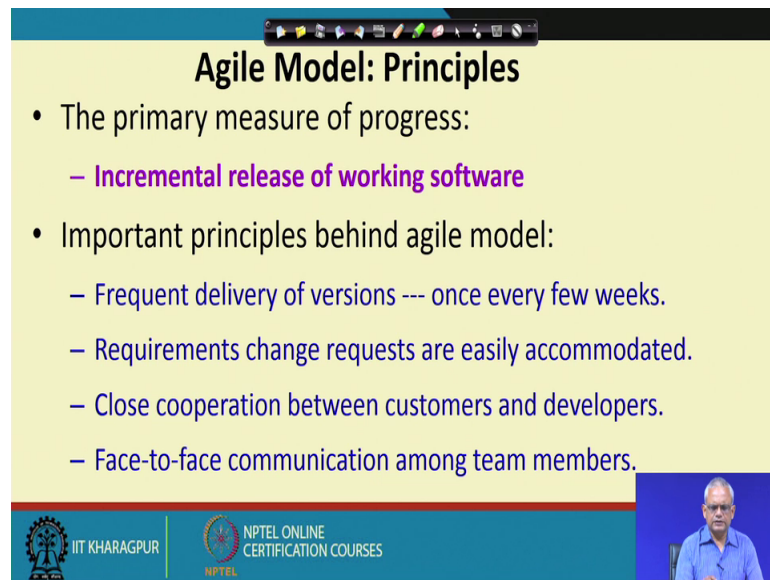
(Refer Slide Time: 18:34)
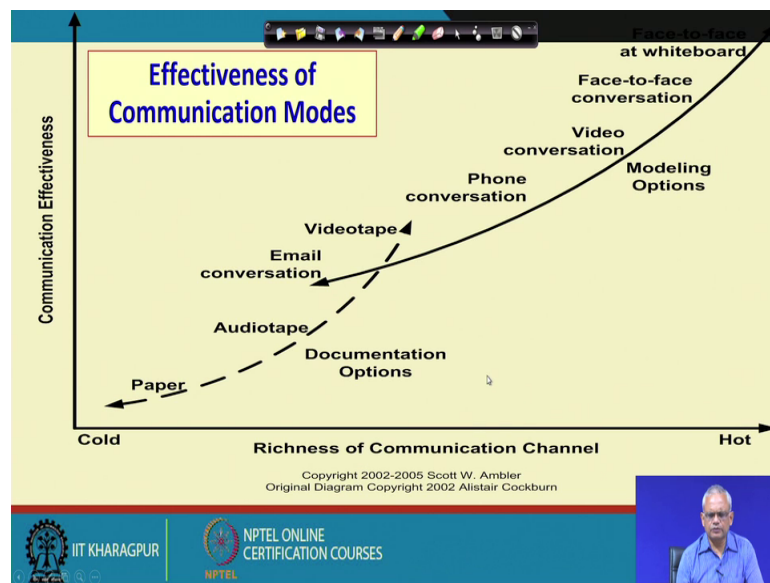
This is evaluation of various communication modes reported by Scott Ambler Alistair Cockburn. The evaluated various communication modes and found that the worst way to communicate is to pass around the paper. Something that may be still little better maybe through audio tape sending out a audio tape. Even email conversation is better video tape may be slightly better, phone conversation is still better, video conversation like Skype etcetera this may be better, but the best form of communication is face to face communication using a whiteboard.

So, write down on the whiteboard if you want to explain anything overall architecture or something, but then the best mode of communication is a whiteboard with face to face communication. And, that is the reason why the agile model it recommends that face to face communication is very important for successful development of software.
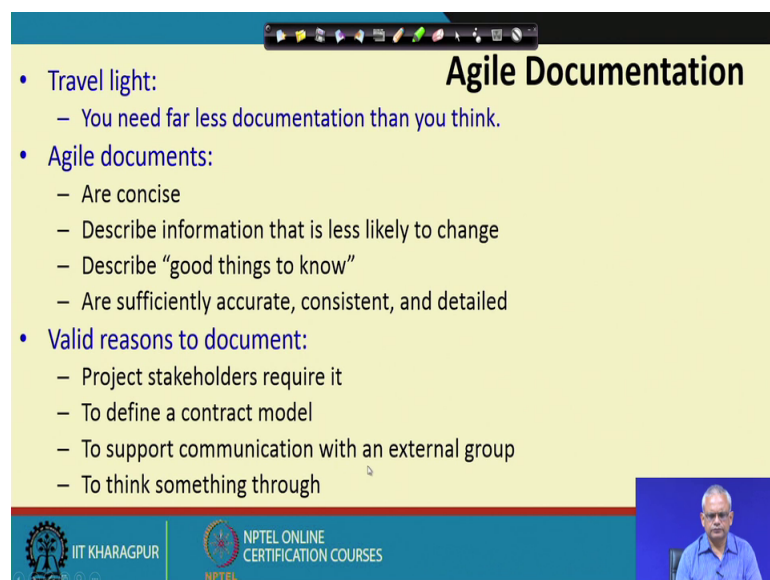
(Refer Slide Time: 19:57)

The software is developed in increments and deployed at the customer site, once every few weeks these are deployed customer feedback is obtained just like the incremental and evolutionary model and these are accommodated, leads to close cooperation between customer and developers, and face to face communication is emphasized among team members.

(Refer Slide Time: 20:31)

But, does agile development model require and documentation to be produced. Yes, documentation can be produced which is very minimal, that is far less documentation

that you think less than that is needed. All documents are to the point very small and the things that are not likely to change are very less likely to change. These are only documented only those aspects which are somebody may be interested to know there will be readers for that document who will benefit by that only those are documented, the document should be accurate consistent and detailed, before we document must identify do we really have to document this.

Valid regional document can be that, it may be useful by the customer. For example, user manual may be to define a contact model. Maybe there is external group we want to get their suggestion and for them we want to prepare a document. And of course, we might document something just to think over it.

(Refer Slide Time: 22:09)
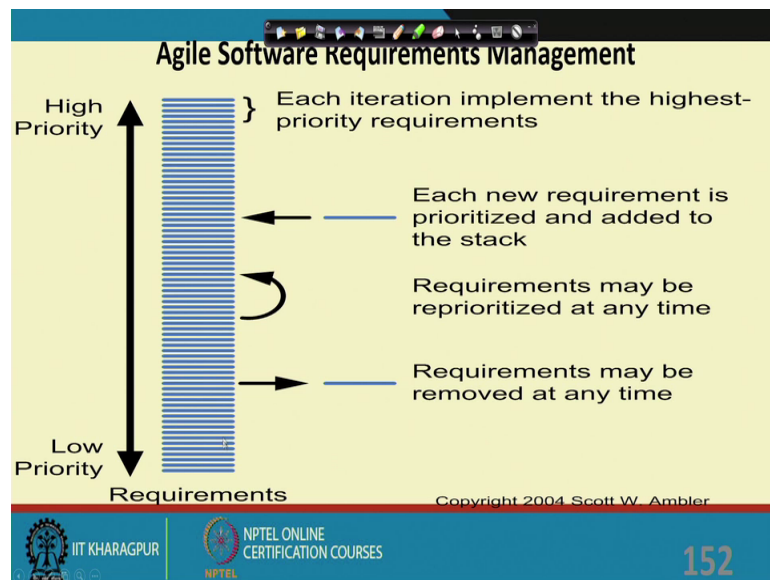
The requirements they keep on coming as is any evolutionary model and this is again taken from Scott ambler, he describes how the requirements are maintained?. So, these are all the requirements and these are maintained in terms of their priority, each time a requirement comes these are put in the appropriate place. Maybe in the excel sheet in a excel sheet the user stories are maintained. Each requirement is actually a user story and at the top of excel are the high priority requirements.

And these are the ones to be implemented first and as the requirements come, they are inserted some requirements may be modified, some requirement may be moved on or load on the list, and some requirements may be deleted based on the customer feedback.

(Refer Slide Time: 23:33)

The agile model has many advantages as you had seen that it helps produce software, least time, least cost, good quality software, accommodates customer requirements, but then there are some problems that we have to keep in mind. The definition of the agile model is rather sketchy, several interpretations may be possible and therefore, high quality people skills are required.

Long term plan long term design etcetera are not made here only focus on one increment. And, also since we keep on getting the customer feedback, it is harder to manage feature creep and customer expectations. And, also up front it would be difficult to quantify the cost time and quality, because we do not know how many increments what features exactly will be there.

(Refer Slide Time: 24:55)

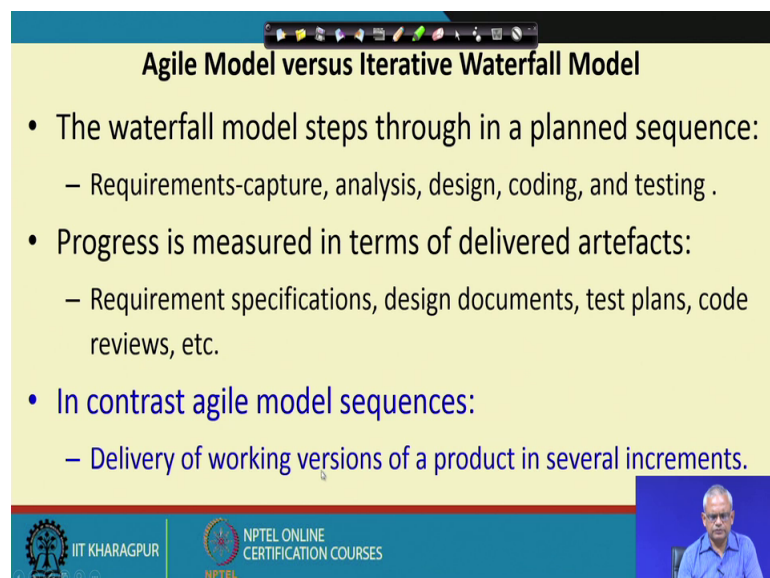The other difficulties are that lot of the knowledge is shared among the developers through verbal communication rather than any formal document. And therefore, after the developers disperse those now knowledge may vanish.

And, also verbal communication can be misinterpreted external review can be difficult to get, and when the project is complete and the team disperses maintenance may become difficult of course, towards the end of the project all the important aspects of the project is documented before the project team disperses.

(Refer Slide Time: 25:49)

Now, let us quickly compare the agile model with various other models. How does the agile model compare with the iterative waterfall model? Waterfall model is a heavy wet process and also the requirements are needed to be identified upfront, and the development proceeds for a planned sequence of activities. The man the manager plans about when requirements will complete analysis, design, coding and testing will complete. And, after each phase documents are produced and the progress is made measured in terms of the delivered at effects.
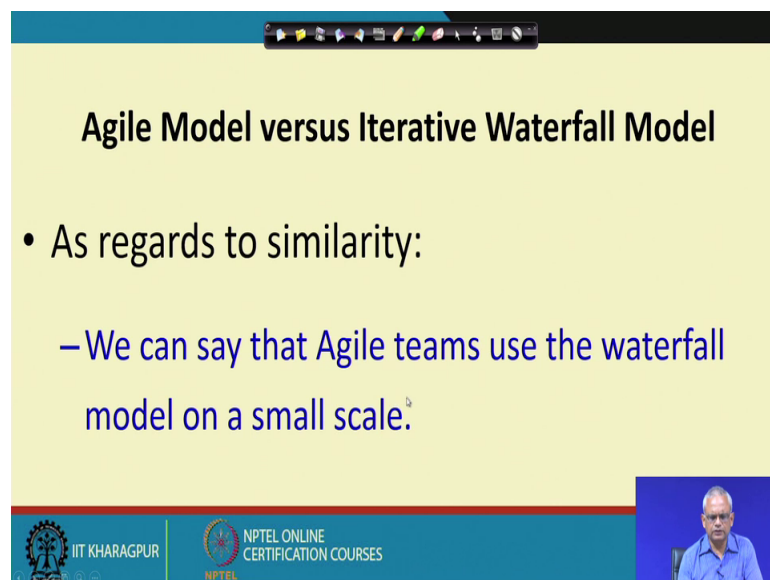
In contrast the agile model the progress is measured in terms of the delivered software. And also initial requirement capture requirements specification etcetera is not there no long term planner made only short term plan for one increment is made.
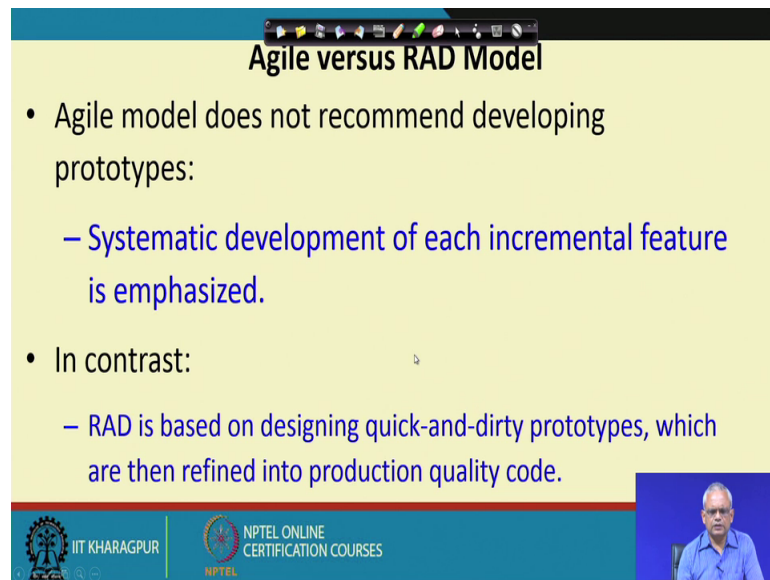
(Refer Slide Time: 27:10)

But, do they have any similarity at all we can say that the agile teams for every increment incremental software they may use a waterfall model, in a small scale. So, over a 2 week or one week period they might develop one increment and that they use waterfall model.

(Refer Slide Time: 27:37)

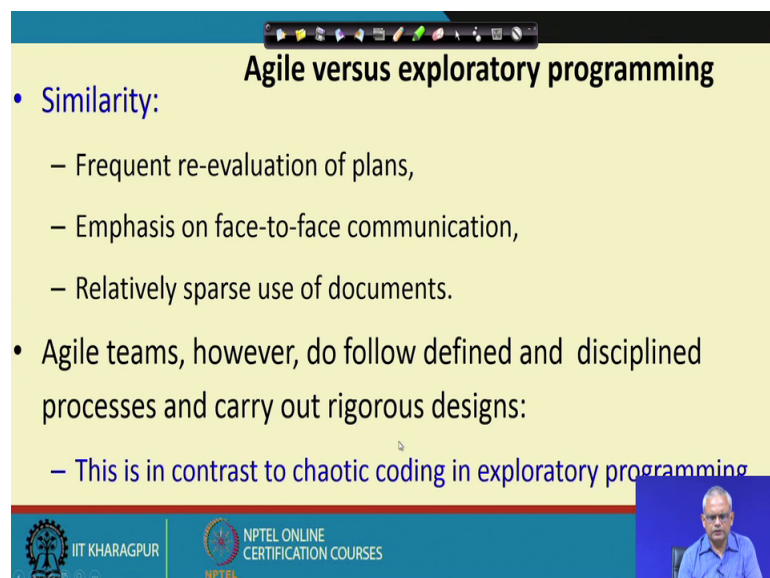How does the agile model compare with a RAD model?

The agile model as we discussed does not recommend developing prototypes and here every iteration or the increment is developed using incremental using a systematic technique. In the contrast in the RAD model quick and dirty prototypes are produced and these are refined into production quality code, but how does the agile model compared with exploratory program, because exploratory programming and agile have many similarities.

(Refer Slide Time: 28:17)

For example, face to face communication, documents are discouraged, frequent revaluation of the plants and so on, but then in contrast to the exploratory model here it is there is a system here, over the increments it is developed every increment is developed according to a plan. And, design, coding, all are according to what for every increment we use waterfall model and this is in contrast to the chaotic coding in the exploratory style.

So, we are nearing the end of this lecture. We will stop here. And, in the next lecture we look at one of the agile development techniques, it is extreme program.

Thank you.