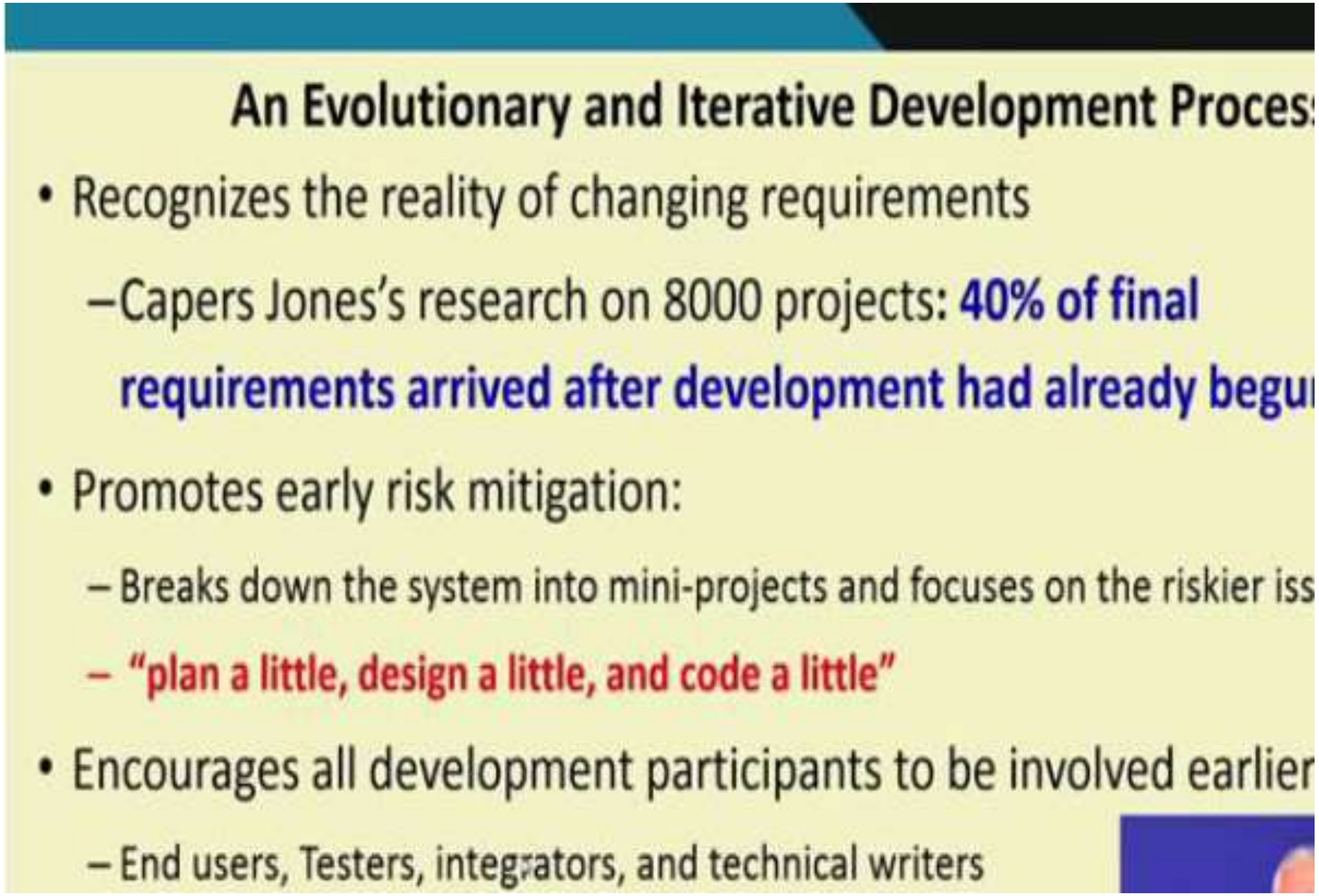
Evolutionary model with iterations

- Over the last few lectures, we had looked at few important software development lifecycle models. We had started looking at the classical model which is very intuitive, the classical waterfall model and the derivatives of this model namely the iterative waterfall model, prototyping model and so on.
- And over the years few more models came up as the shortcomings of the waterfall model were noticed. These were the RAD model, the incremental model, incremental with iteration and the evolutionary model.
- In the last lecture, we had looked at the incremental model.
- Today, we will first look at the evolutionary model and then, we will look at the agile development model.
- **the term iteration is used when existing functionality which was developed and delivered is iterated and refined b**ased on customer feedback



An Evolutionary and Iterative Development Proces

- Recognizes the reality of changing requirements
  - Capers Jones's research on 8000 projects: **40% of final requirements arrived after development had already begu**
- Promotes early risk mitigation:
  - Breaks down the system into mini-projects and focuses on the riskier iss
  - **"plan a little, design a little, and code a little"**
- Encourages all development participants to be involved earlier
  - End users, Testers, integrators, and technical writers

- One of the main problem with the waterfall based model is that the requirements are defined and frozen at the start of the project.

- But the practical situation is quite different . Capers Jones researched on 8000 projects and he found that 40 percent of the requirements were arrived, when the development had already begun.

- If we had frozen the requirements, the start; then, this 40 percent changes will be very difficult to incorporate. Because in reality this is the average figure; 40 percent, it can even be more for some projects.

- Therefore, there is need for a model which can effectively handle requirement changes.

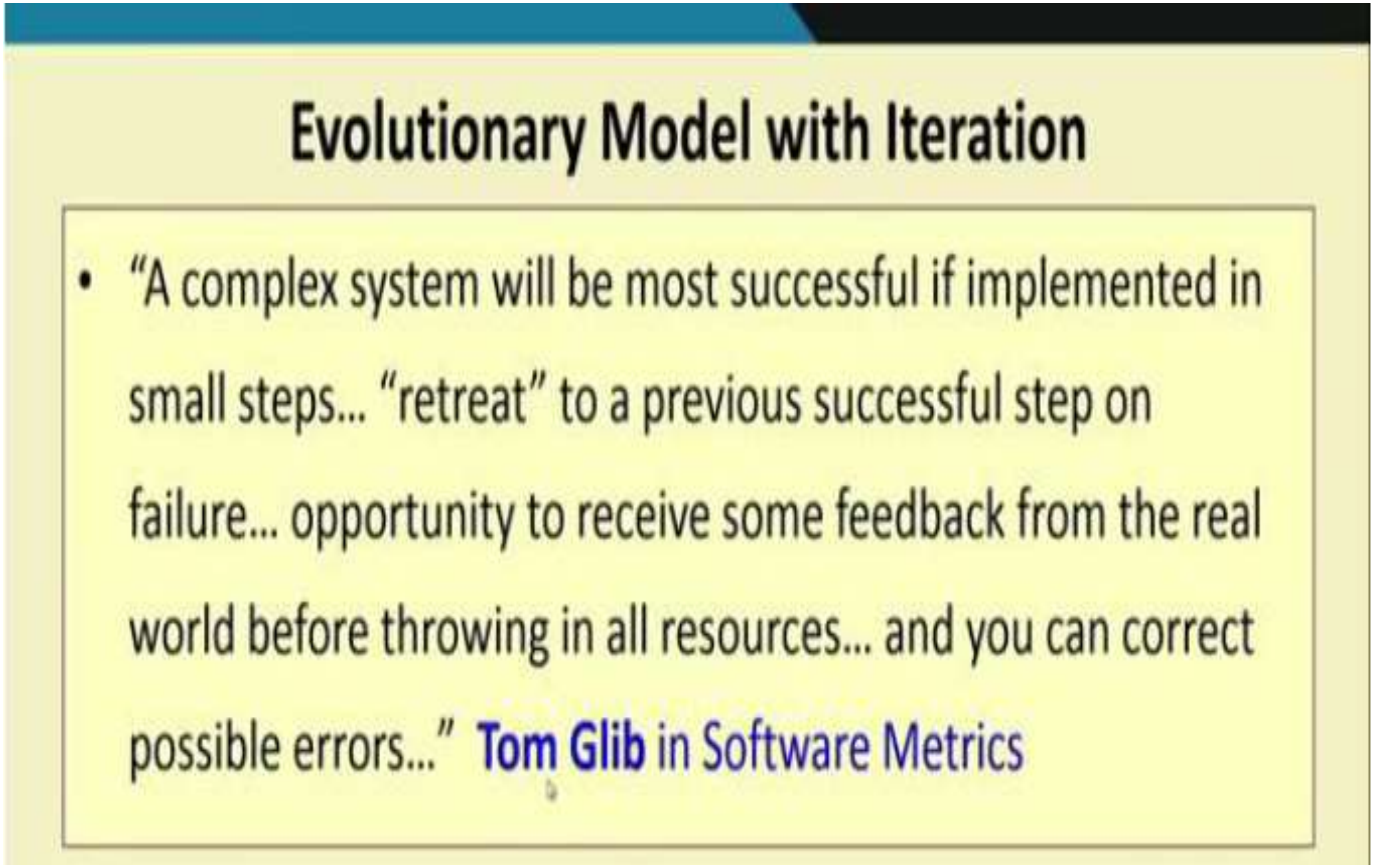
- Requirements change can be due to various reasons

- Business: Business changes very rapidly. Once they have defined a business procedure may be after a month the business procedure changes

- Technology : maybe the technology changes
- or maybe
- Misunderstood requirements : developers might have not understood what is required or the customer might have told something else, maybe they had forgotten to tell something and so on

- So we need a model which can effectively handle requirement changes.

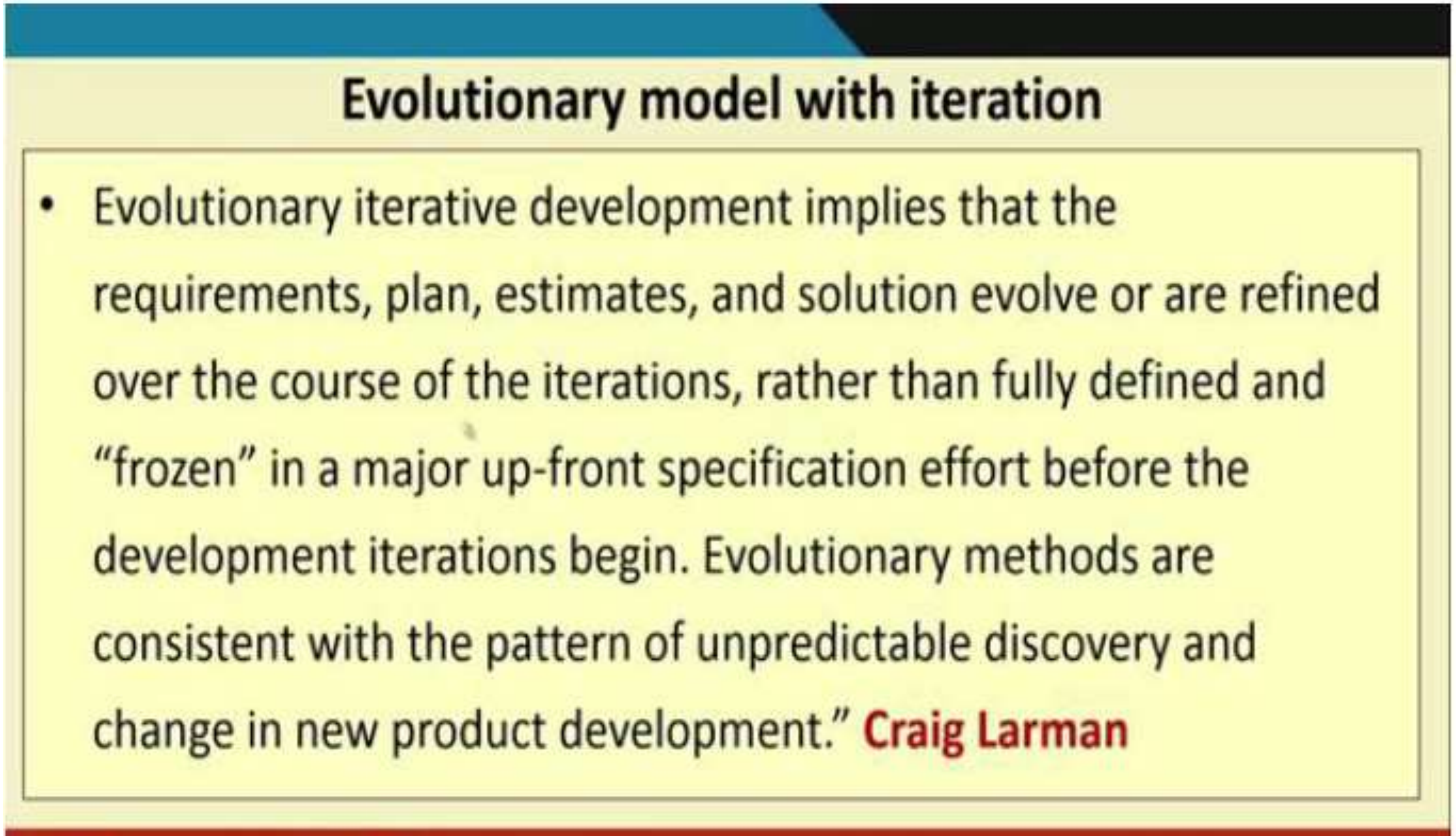
- *One way that has been considered very promising is that each time do only small part of the system. Start with those that are riskier and likely to change and the customer is not sure about. Plan a little for this small work, design a little, code a little and once the small part is completed, give it to the customer for his evaluation and feedback. Here the customer is encouraged to participate. The end user the tester, integrator, technical writer all are involved in the development.*



Evolutionary Model with Iteration

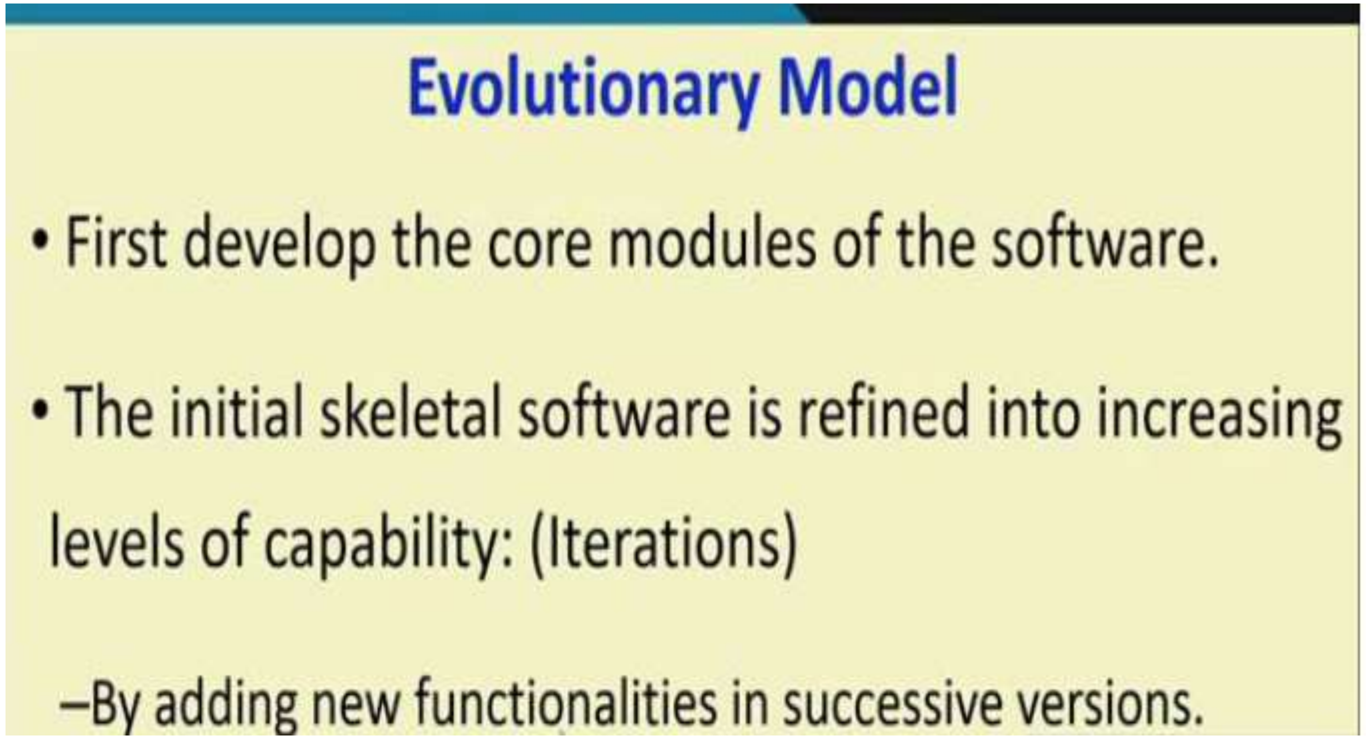
- "A complex system will be most successful if implemented in small steps… "retreat" to a previous successful step on failure… opportunity to receive some feedback from the real world before throwing in all resources… and you can correct possible errors…" **Tom Glib** in Software Metrics

- So, according to Tom Glib, the evolutionary model which he captures here in this few lines is a very promising model to handle changes to the requirement due to various reasons.



Evolutionary model with iteration

- Evolutionary iterative development implies that the requirements, plan, estimates, and solution evolve or are refined over the course of the iterations, rather than fully defined and “frozen” in a major up-front specification effort before the development iterations begin. Evolutionary methods are consistent with the pattern of unpredictable discovery and change in new product development.” **Craig Larman**



Evolutionary Model

- First develop the core modules of the software.
- The initial skeletal software is refined into increasing levels of capability: (Iterations)
  - –By adding new functionalities in successive versions.

- But then, the question is that how is the incremental development and the evolutionary development differ?

- Because as you can see that in both these models incremental and evolutionary, there are small increments over which the system is developed and these increments are deployed at the customer site and feedback obtained
- In the purely incremental model as you might have remember from our last discussion that all those requirements are identified and these are planned into small increments. Each increment is developed deployed feedback obtained and changes can happen. In an incremental model, all the requirements are gathered to start with whereas, in evolutionary model, it is not the case. Here after an overall understanding of the system we do not really capture all the requirements, but then start with some of the core or riskier modules.

- In the evolutionary model, the requirements are not frozen.

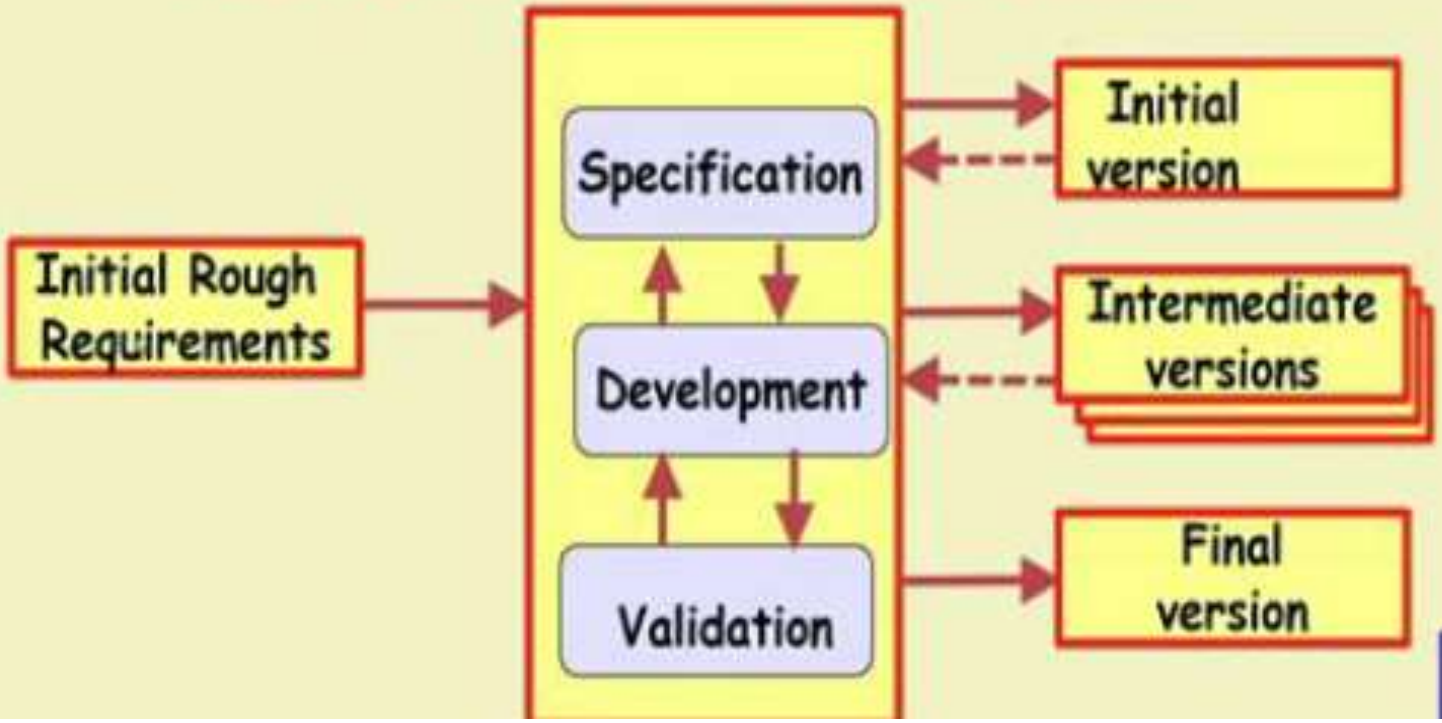
- Here the customer is encouraged to participate, to give feedback based on which the further developments occur and the customer feedback is taken into account and functionalities get modified.

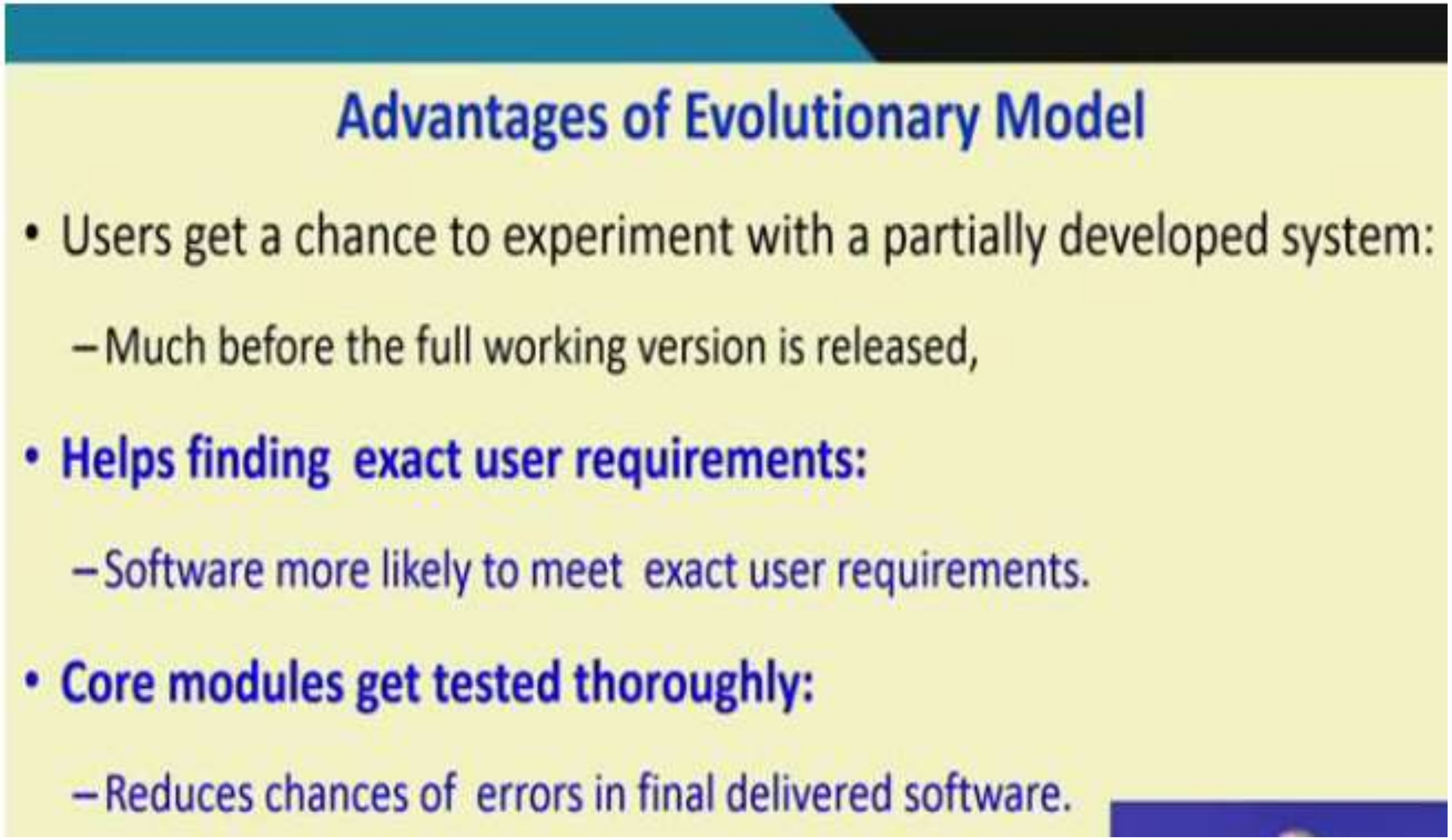
- So, here the requirements are modified in a iteration, some of the existing functionalities as well as new functionalities are delivered

- Here, successive versions are developed and deployed at the cost customer site. Each version is capable of performing some useful work which the customer can use and give feedback and a new release can include a new functionalities and also existing functionalities may get modified and refined

Evolutionary Model

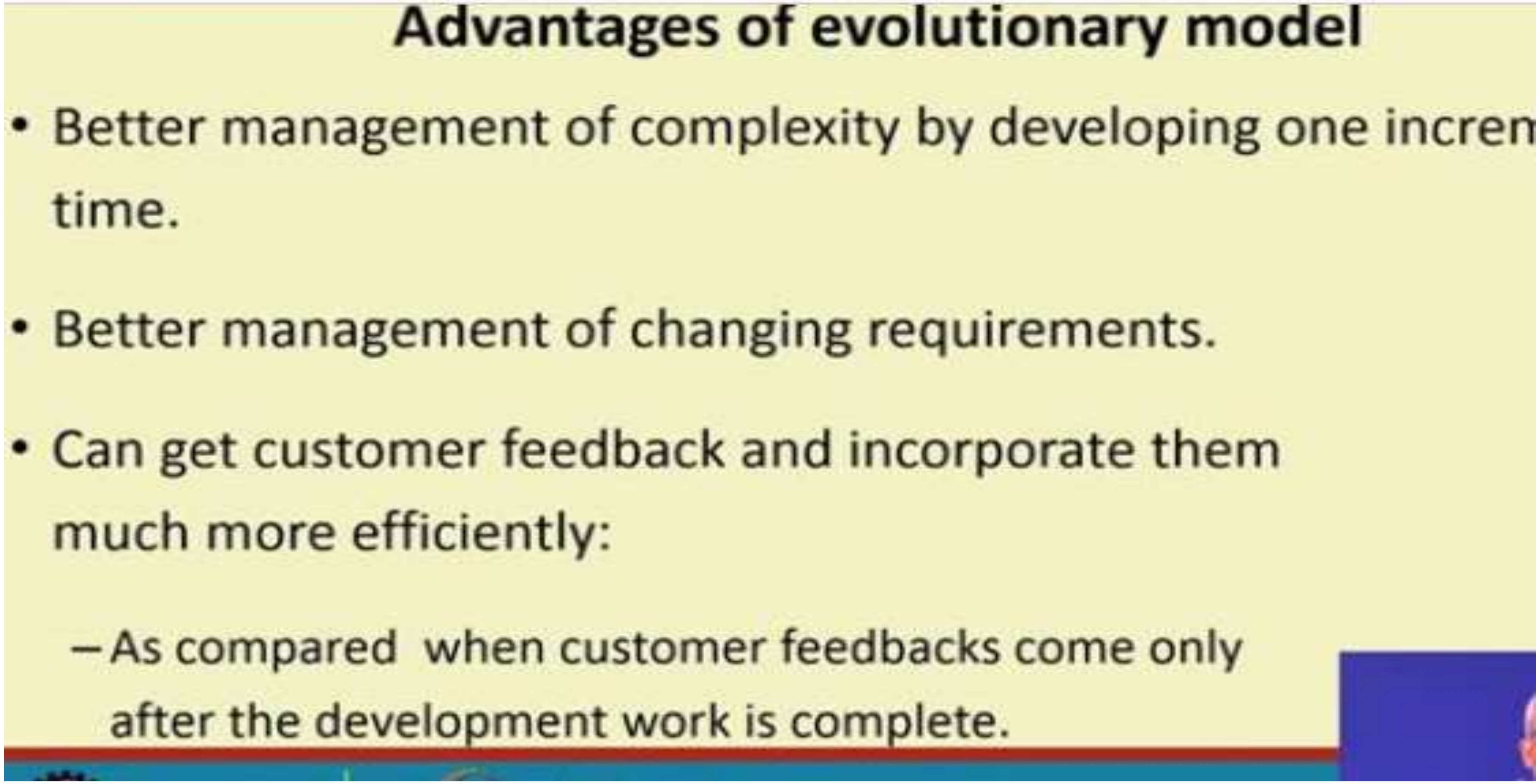
- Evolves an initial implementation with user feedback:
  - Multiple versions until the final version.





Advantages of Evolutionary Model

- Users get a chance to experiment with a partially developed system:
  - Much before the full working version is released,
- **Helps finding exact user requirements:**
  - Software more likely to meet exact user requirements.
- **Core modules get tested thoroughly:**
  - Reduces chances of errors in final delivered software.



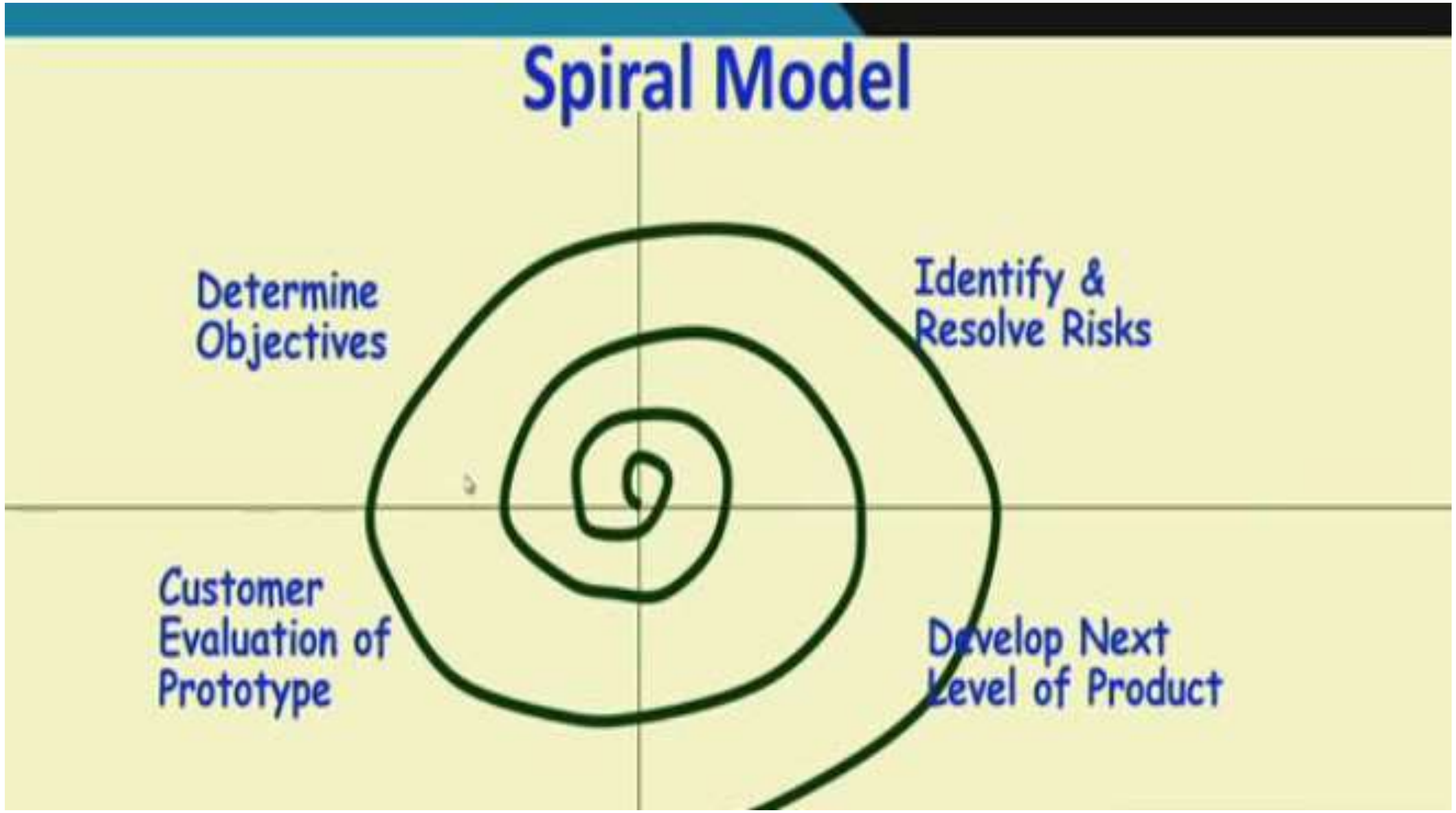
Advantages of evolutionary model

- Better management of complexity by developing one increm time.
- Better management of changing requirements.
- Can get customer feedback and incorporate them much more efficiently:
  - As compared when customer feedbacks come only after the development work is complete.

Problems with the evolutionary model

- You had seen that it is lot of advantages, but then here the process is unpredictable that is each time, we develop a increment deploy and get the customer feedback.
- Now, what if the customer just keeps changing the requirements; keeps on asking for modifications and so on. Therefore, the process is unpredictable. We don’t know when it will finish, because the customer may keep on changing the requirements and also if we have a long term plan, then it becomes easier to deploy manpower, recruit them, schedule work monitor and so on. Here all these are problematic.
- The system is poorly structured because there is no overall design made its only small parts that are designed and integrated. Since, the code is changed continually, the code structure degrades and the system may not even converge to a final version if each time the customer keeps on asking for changes.

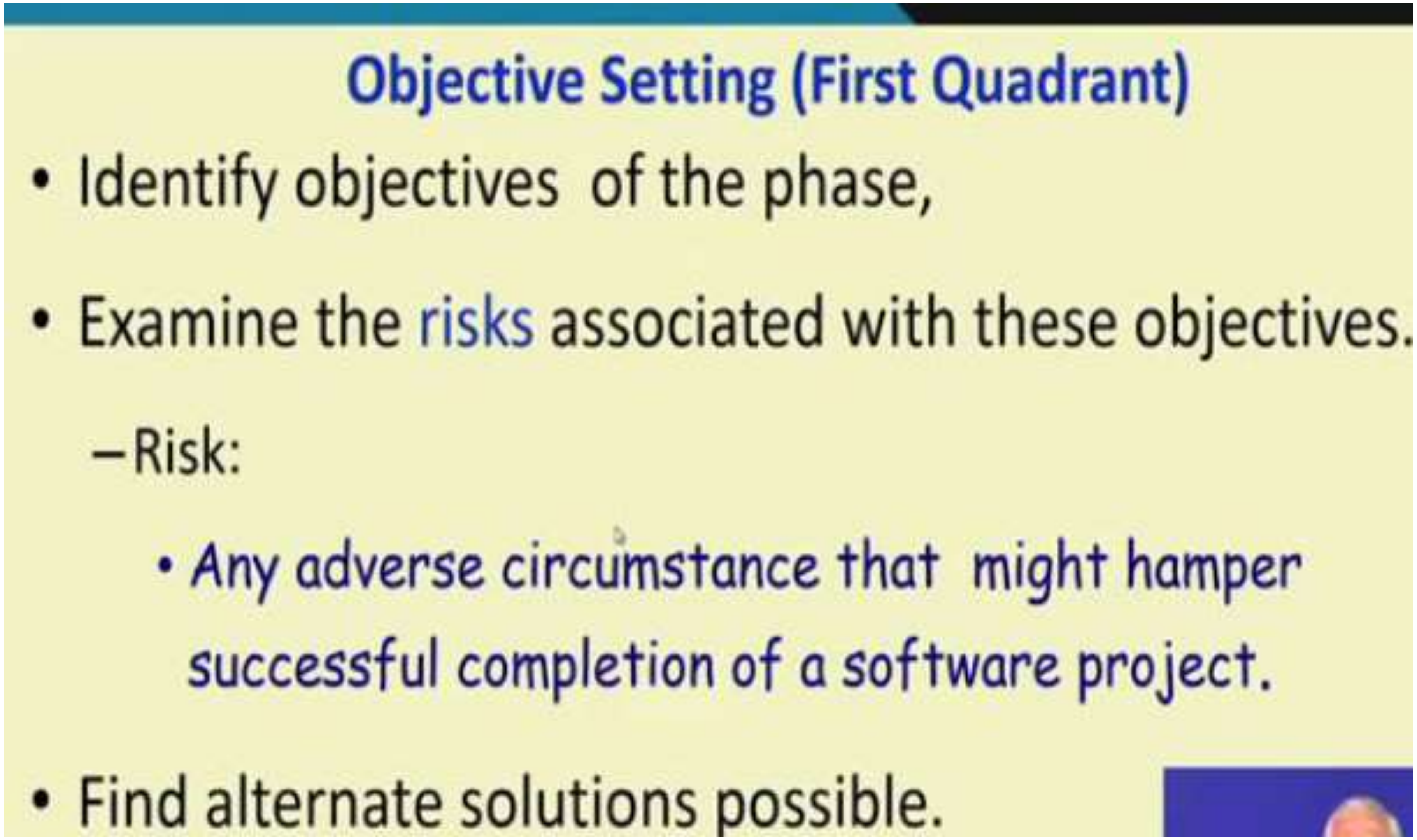
- The customer gets version quickly. They get trained on using the software. If there are bugs, these are fixed quickly in the next iteration.



Spiral Model

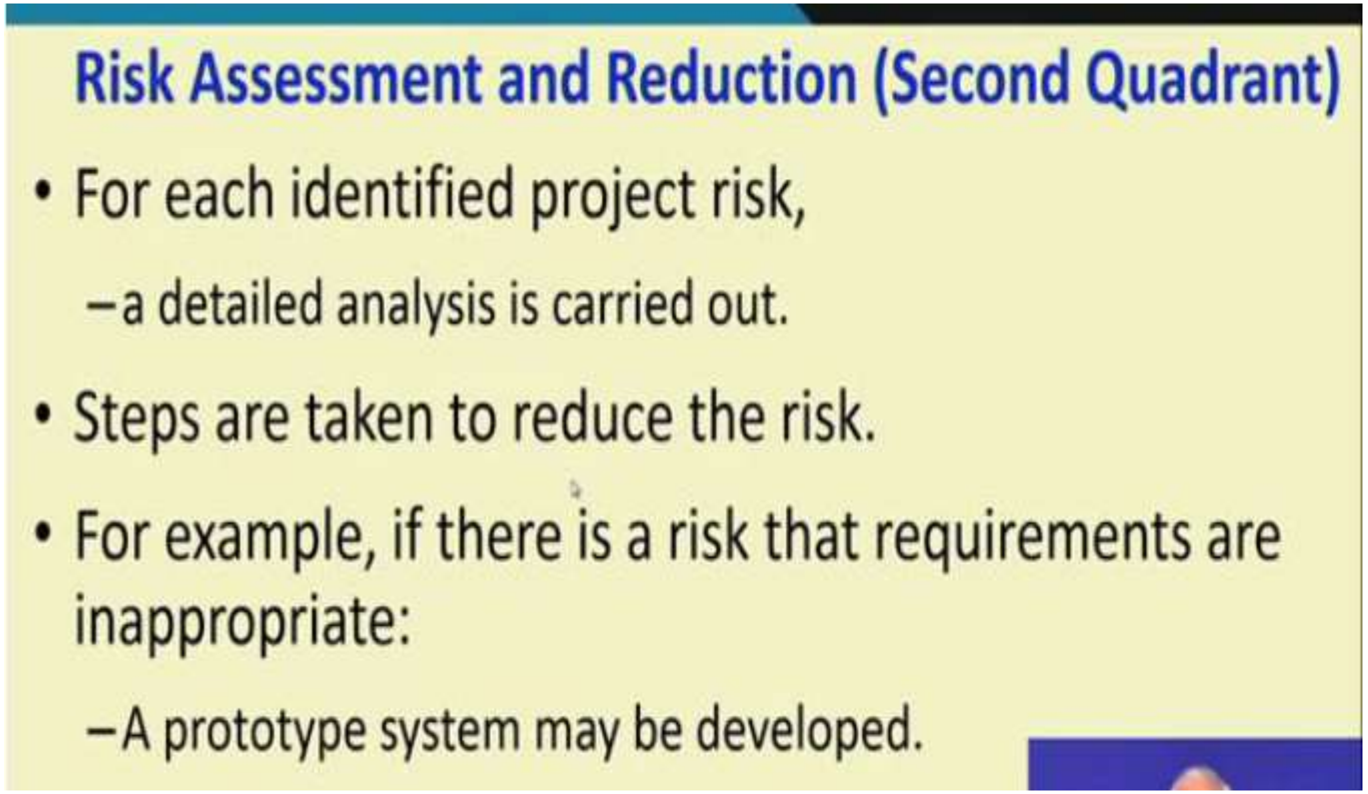
Determine Objectives
Identify & Resolve Risks
Customer Evaluation of Prototype
Develop Next Level of Product

- There are four quadrants on the spiral model, the first quadrant what needs to be done in that phase is identified and then, if there are any uncertainties. Technical uncertainties or the user requirements are not clear. A prototype is developed to resolve the risk and then, the next iteration is developed given to the customer for his evaluation and then, another feature is taken up.
- So, over each iteration one or more features are taken up. The risk is resolved and then the next development occurs and customer feedback.
- As can be seen here, this model is ideally suited for very risky projects, where the technology is not known. It is possibly the first time that this kind of software is being developed. The technology is not certain and so on. So, each time some features the most riskiest feature is identified. Risk is resolved, it is developed and given for customer feedback and the next set of features are taken up.



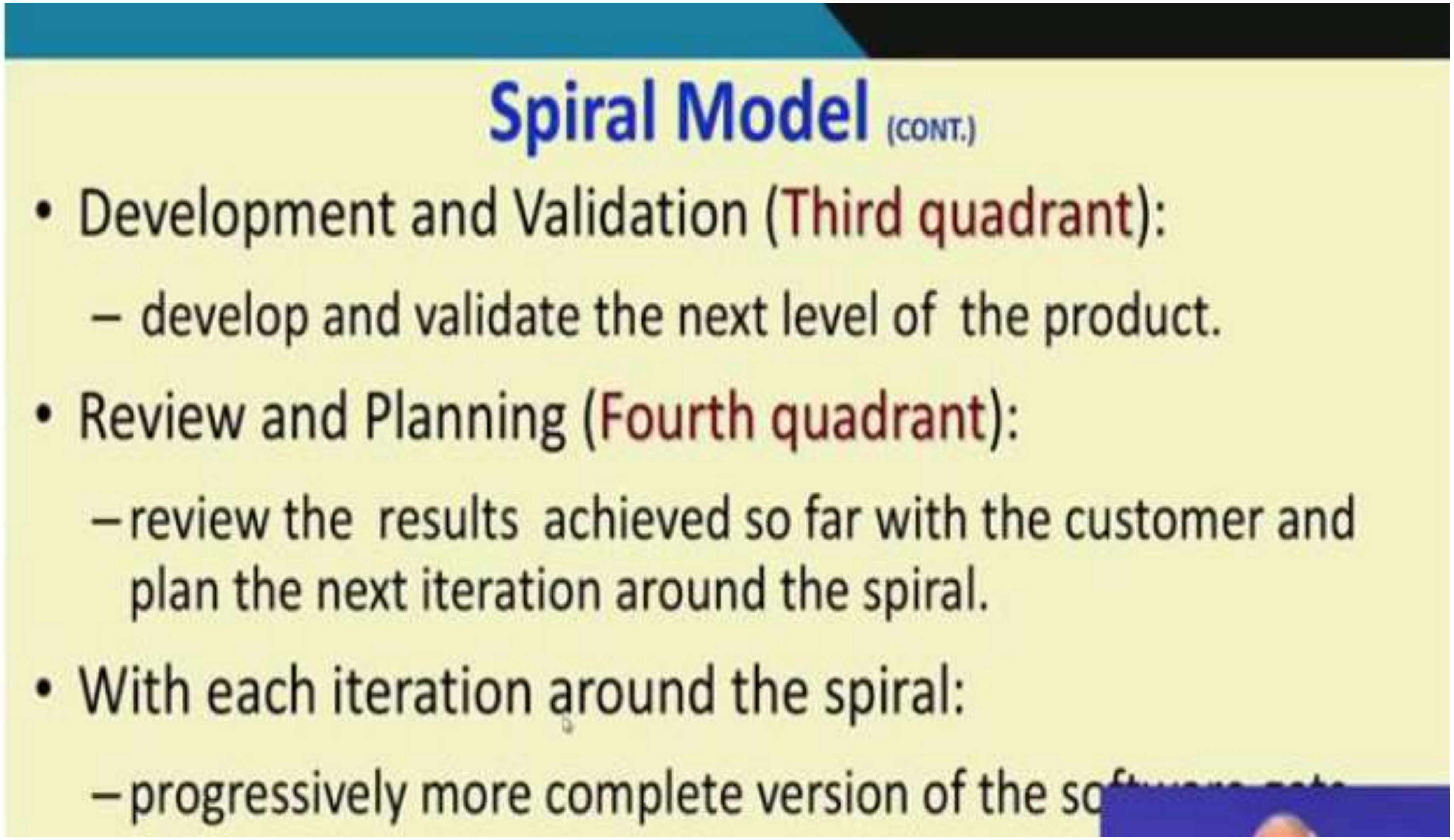
Objective Setting (First Quadrant)

- Identify objectives of the phase,
- Examine the risks associated with these objectives.
  - Risk:
    - *Any adverse circumstance that might hamper successful completion of a software project.*
- Find alternate solutions possible.



Risk Assessment and Reduction (Second Quadrant)

- For each identified project risk,
  - a detailed analysis is carried out.
- Steps are taken to reduce the risk.
- For example, if there is a risk that requirements are inappropriate:
  - A prototype system may be developed.



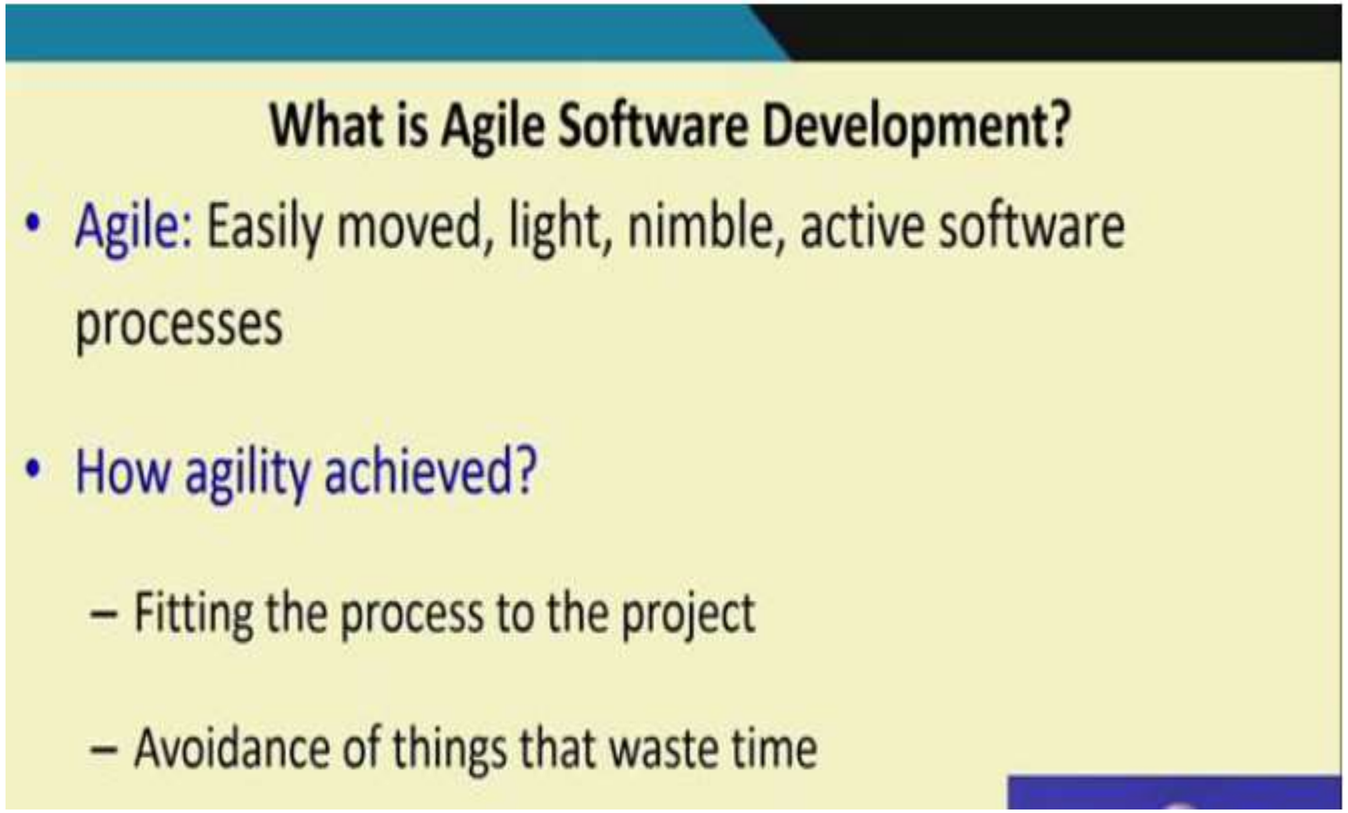
Spiral Model (CONT.)

- Development and Validation (Third quadrant):
  - develop and validate the next level of the product.
- Review and Planning (Fourth quadrant):
  - review the results achieved so far with the customer and plan the next iteration around the spiral.
- With each iteration around the spiral:
  - progressively more complete version of the so

- The spiral model is also called as a Meta model because a single loop of the spiral is a waterfall model, there are iterations and therefore, it incorporates evolutionary and incremental models. It uses prototyping.
- The stepwise approach of the waterfall model is retained and therefore, this model can be degenerated or can be used as any other model.

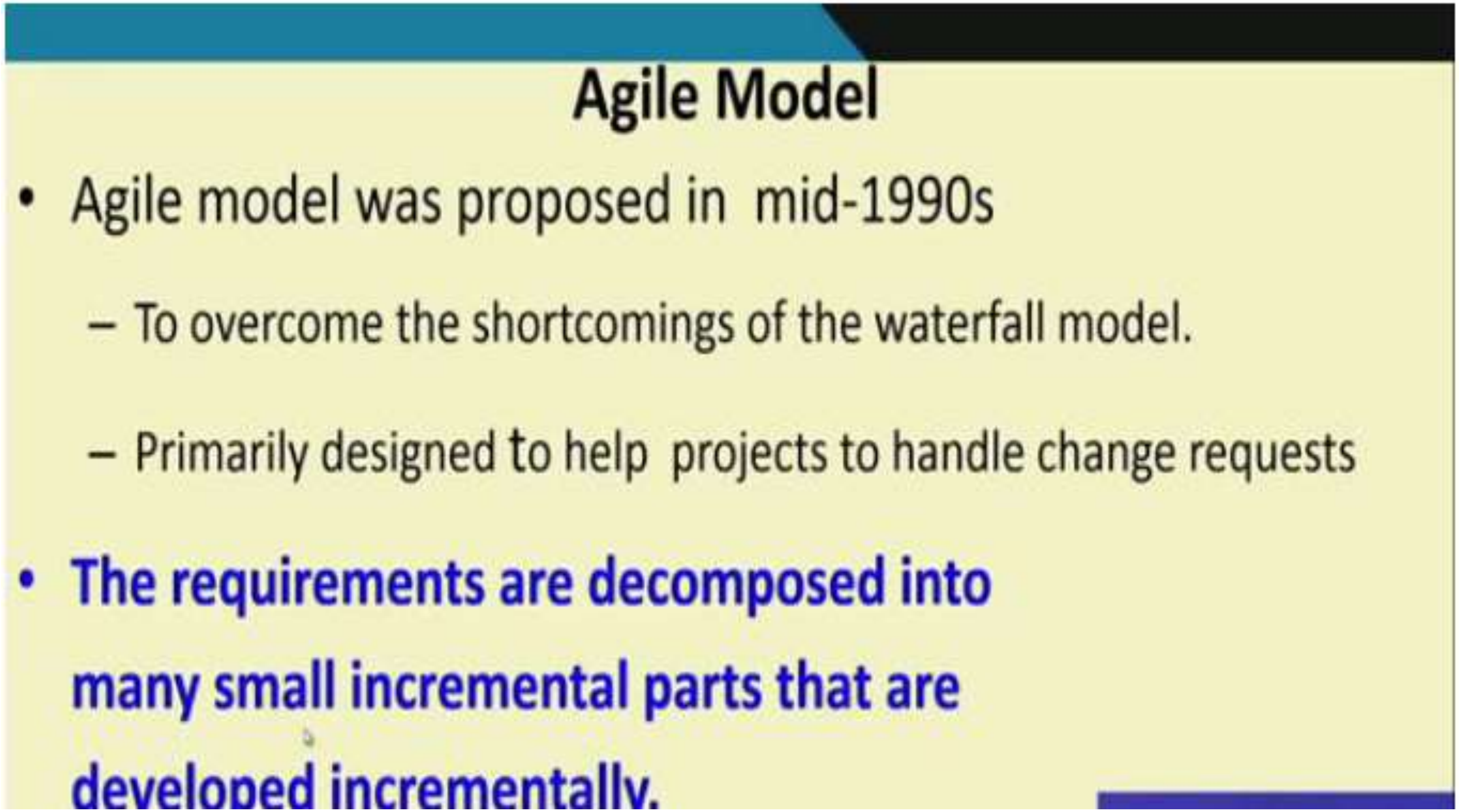
- But then, need to remember that the spiral model is suitable for very risky projects and large projects.

What is Agile Software Development



What is Agile Software Development?

- Agile: Easily moved, light, nimble, active software processes
- How agility achieved?
  - Fitting the process to the project
  - Avoidance of things that waste time



Agile Model

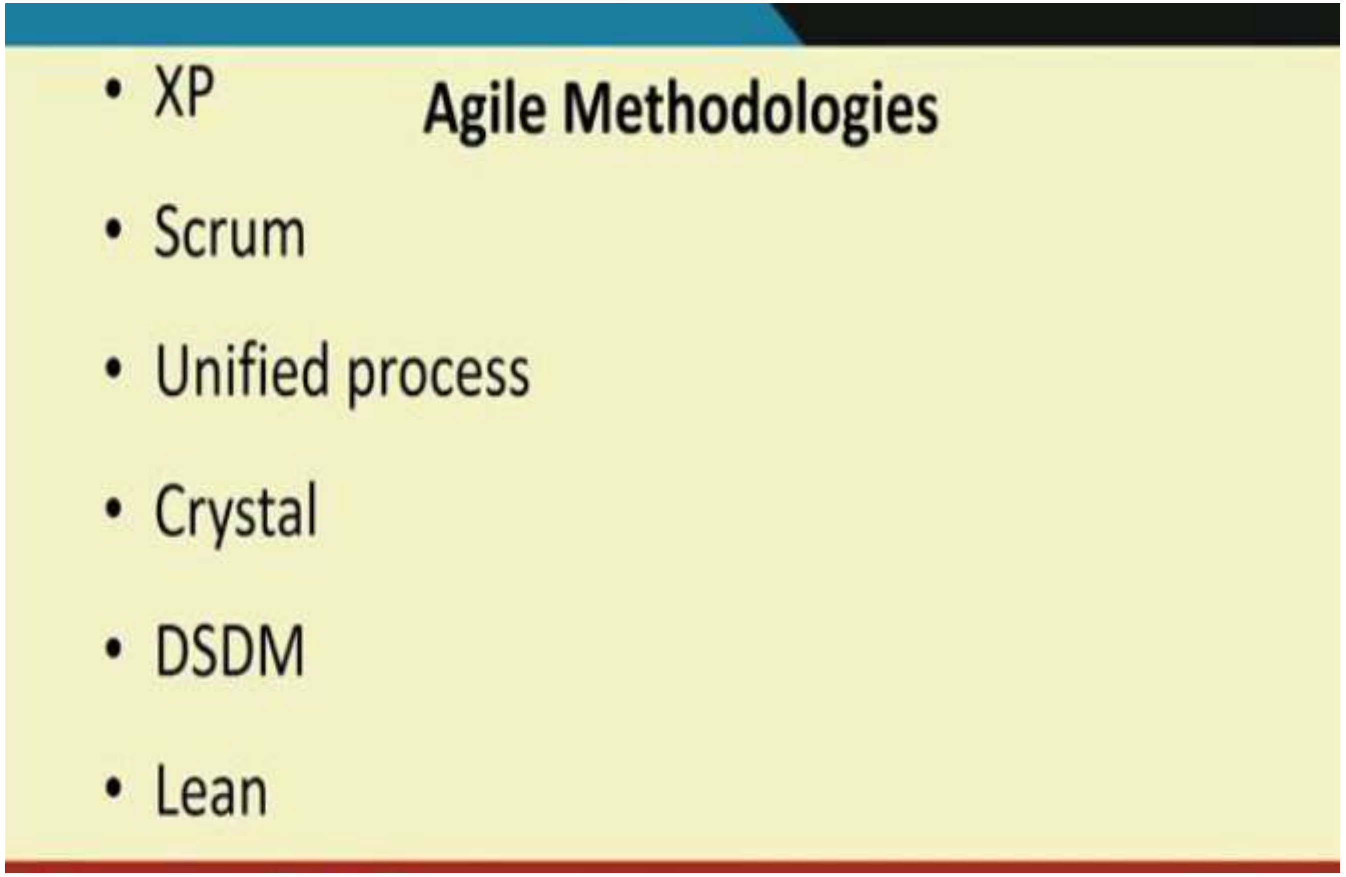
- Agile model was proposed in mid-1990s
  - To overcome the shortcomings of the waterfall model.
  - Primarily designed to help projects to handle change requests
- **The requirements are decomposed into many small incremental parts that are developed incrementally.**

Salient Features of the Agile Model

- There are a few things which are important here.

1. Instead of producing elaborate documents, following process rigorously here the individuals and they interact with each other rather than passing on a document to each other. They explain to each other through interaction. In a waterfall model, the progress of the development is measured in terms of the documents produced; is the requirement document complete; is the design document complete; is the detailed design complete; is the code documentation complete; test documentation complete. But here, the progress is measured in terms of the working software that has got developed. How many iterations have been completed and each iteration, some working software is deployed at the customer site

- 2. Here, the customer is encouraged to participate in development, is encouraged to collaborate in the development rather than just signing a contract at the start of the project and here, it is assumed that changes will occur, the requirements will change and how to handle those rather than making a long term plan and following that plan, here changes are welcome and how to handle those changes those are important issues here



Agile Methodologies

- XP
- Scrum
- Unified process
- Crystal
- DSDM
- Lean