


**Software Engineering**  
**Professor Doctor Sridhar Iyer**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Bombay**  
**Doctor Prajish Prasad**  
**FLAME University**  
**Software Development Models- Agile Perspective**

(Refer Slide Time: 0:27)

## Recap

- Software development lifecycle
  
- Different models in the plan and document perspective -
  - Waterfall
  - Prototype
  - Spiral



**IIT Madras**  
B.S. Degree

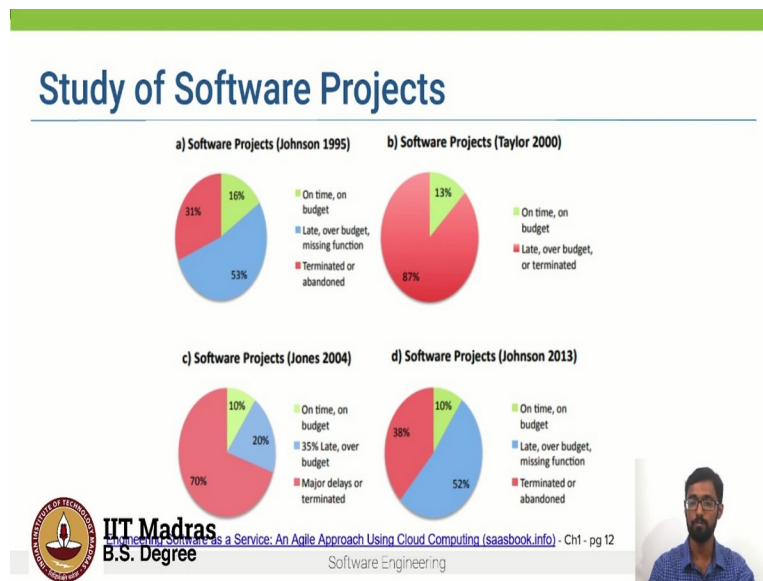
Software Engineering



In the previous videos, we looked at the software development lifecycle, particularly the waterfall model and its extensions. And these models are known as the plan and document process models, wherein you come up with a plan for the project, which includes extensive and detailed documentation of all the phases of that plan. And progress of software development is measured against this plan.

We saw different models like the waterfall model, the prototype model, the spiral model, etc. And although these planning document processes, they brought discipline to software development, they were not very effective in terms of on time delivery, and being within the specified budget. So, let us look at some examples.

(Refer Slide Time: 1:29)



So, the set of images on the screen, it shows the status of software projects across different years from 1995 to 2013. And what these series of studies show is that 80 to 90 percent of all software projects are either late or over budget or simply just abandoned. And this is a shocking number. So, on the other hand, it is just 10 to 15 percent of software projects which are on time and on budget. So, this was considered a severe drawback of the plan and document perspective.

(Refer Slide Time: 2:25)

### Agile Manifesto

- 4 key principles -
  - Individuals and interactions over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan

IIT Madras B.S. Degree  
Software Engineering

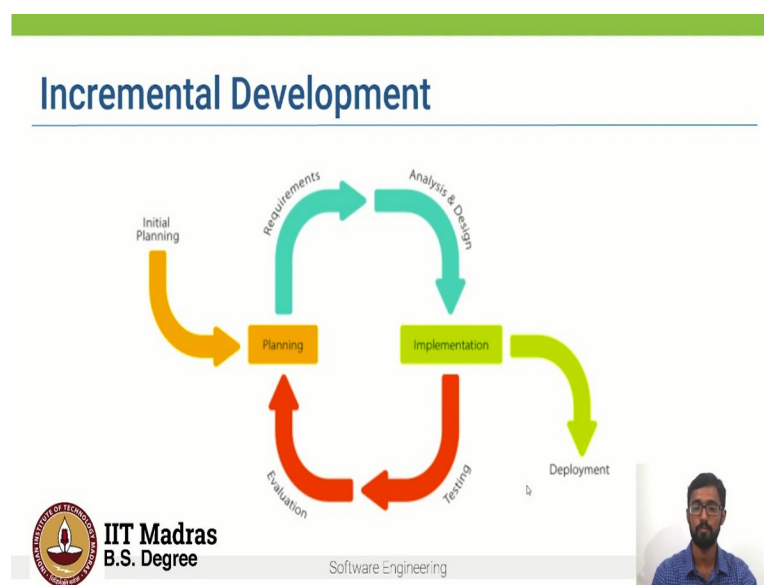
So, to address these problems or these difficulties, around 20 years ago, in February 2001, a group of software developers then met to develop a lighter weight software development lifecycle. And this came to be known as the Agile Manifesto. So, this manifesto the Agile

Manifesto is built on four key principles. One, it emphasizes individuals and interactions over processes and tools. So, although processes and tools are important, it emphasizes interactions between individuals in the development team as well as with the clients.

Second, rather than focusing a lot on comprehensive and extensive documentation, the emphasis is on delivering working software. And this is delivered in increments. And the Agile Manifesto emphasizes a lot on collaboration with the customer to actually understand the exact needs of the customer, rather than negotiating over contract details.

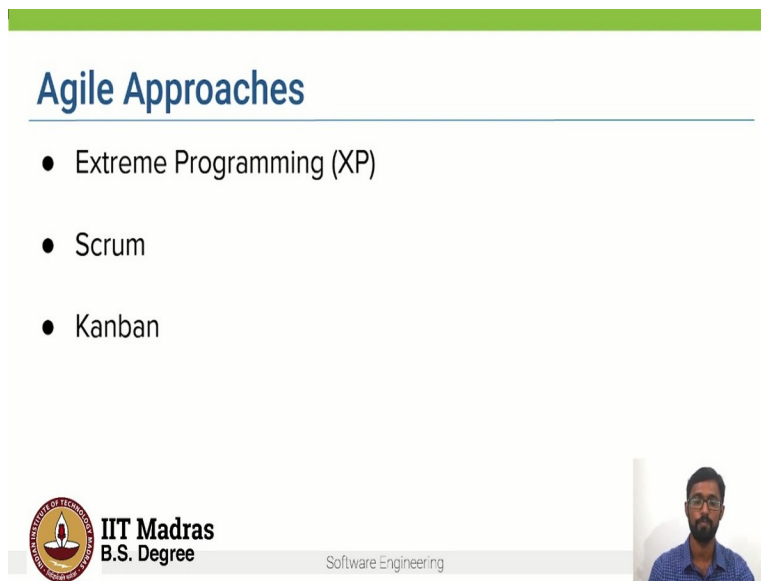
And finally, instead of following a strict and rigorous plan, agile emphasizes on responding to change whenever it is required. And what the manifesto states is that although the items on the right are valuable, the items on the left are more valuable in the Agile development process.

(Refer Slide Time: 4:25)



This agile process is being widely used today. And in this approach software projects are built using multiple iterations and thus it is known as an iterative software development model or incremental software development where the teams work together to deliver the product in small increments. So, instead of waiting till the end to deliver the entire product teams, they develop prototypes of key features and quickly release the prototype for feedback. For example, in the first iteration, the development team might build features for key requirements, they then show it to the client and get their feedback. So, even if the client wants changes, the team can quickly refine it in the next iteration.


(Refer Slide Time: 5:24)




The slide features a green header bar at the top. Below it, the title "Agile Approaches" is written in a blue font and underlined. A bulleted list follows, containing three items: "Extreme Programming (XP)", "Scrum", and "Kanban". At the bottom left is the IIT Madras logo and text "IIT Madras B.S. Degree". At the bottom right is a small video feed of a man with a beard and glasses, wearing a blue shirt. The text "Software Engineering" is centered at the bottom.

## Agile Approaches

- Extreme Programming (XP)
- Scrum
- Kanban

 **IIT Madras**  
B.S. Degree

Software Engineering




So, let us look at some agile approaches. So, one of the earliest agile approaches is extreme programming or XP. And it has several key practices like behavior driven design, test driven development, pair programming etc. We will look at some of these in detail in the upcoming weeks. Another approach is known as Scrum, in which the product is built in a series of iterations known as Sprints, which are roughly one to two weeks long. And it this helps break down a big complex project into several smaller bite sized pieces.

Another approach is known as Kanban, where the software to be built is again divided into small work items. And these are represented on a Kanban board, allowing team members to see the state of every piece of work at any given time. So, in this course, we will be following several practices from extreme programming and Scrum such as behavior driven design, test driven development, sprint and all for our group projects. .

(Refer Slide Time: 6:50)


## Agile Philosophy

- Rather than just following approaches and practices - more of adhering to the broad philosophy
  - Individuals and interactions over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan



**IIT Madras**  
B.S. Degree

Software Engineering




So, the important thing to note here is that rather than just following these approaches and practices, the Agile development is more of adhering to the broad philosophy and the principles outlined by the Agile process. So, when you approach software development in this manner, it is generally good to live by these values and principles, and use them to help figure out the right things to do given your particular context.


(Refer Slide Time: 7:33)

## Reflection Spot

Plan and Document vs Agile - When to use?




Please pause the video and written down your responses



**IIT Madras**  
B.S. Degree

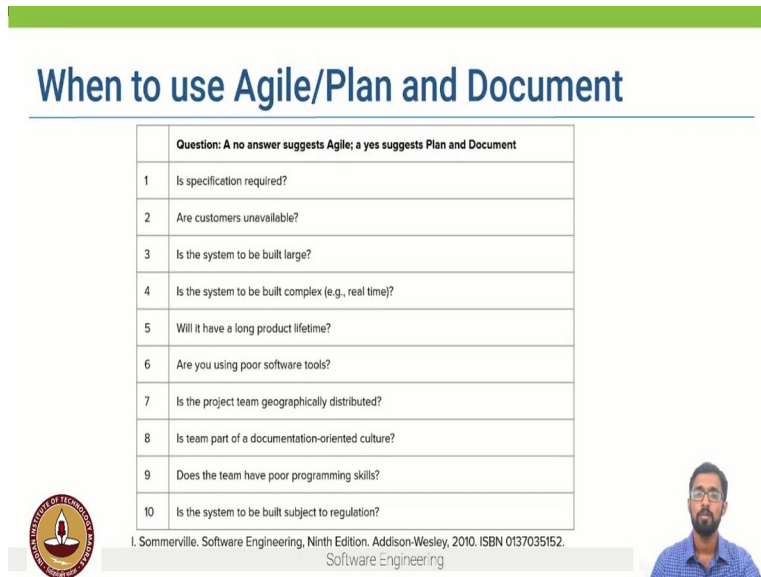
Software Engineering



So, now that we have seen these two broad perspectives, the planning document perspective and the Agile perspective, let us reflect on this question. When should we use the planning document perspective? And when should we use the Agile perspective? So, let us say we want to build a software product, which process should I use? What are certain factors or

characteristics, which will help you determine which model to use? So, please pause this video and think about your responses before proceed.


(Refer Slide Time: 8:26)



**When to use Agile/Plan and Document**

	Question: A no answer suggests Agile; a yes suggests Plan and Document
1	Is specification required?
2	Are customers unavailable?
3	Is the system to be built large?
4	Is the system to be built complex (e.g., real time)?
5	Will it have a long product lifetime?
6	Are you using poor software tools?
7	Is the project team geographically distributed?
8	Is team part of a documentation-oriented culture?
9	Does the team have poor programming skills?
10	Is the system to be built subject to regulation?

I. Sommerville. Software Engineering, Ninth Edition. Addison-Wesley, 2010. ISBN 0137035152.  
Software Engineering



So, which process model to use depends on several factors. And this set of questions can help us get a better idea of which model to use. So, a no answer to most of these questions suggests agile, whereas if it fits a yes to most of these questions, then a plan and document perspective might be better. So, let us look at these questions.

So, firstly, if the requirements are the specifications, if they are required to be fixed upfront at the start of the project, then a plan and document perspective might be better. It also depends on the clients or their customers. If they are unavailable for most of the time, then maybe a plan and document perspective might be better.

And the decision also depends on the characteristics of the system to be built. So, if the system to be built is large if it is very complex. So, for example, you have to build software for maybe an air missile system or a nuclear power plant there is no margin for error. And hence, extensive planning and documentation is a necessity in such cases. So, it might be better to go for a plan and document perspective.

Another characteristics is about the team, your software team. So, if they are geographically distributed, if the team already is familiar with a documentation model, then it might be better to use a plan and document process and finally, is the system which we have to build is subject to several regulations for example, if you want to build a banking software, you need

approvals from banks from the government etc. hence upfront planning and documentation is necessary.

So, the key thing to remember here is that the type of approach you follow it depends on a lot of factors, it depends on the clients, the characteristics of the system and the team. So, once you start working in a software company, different teams and different organizations will use different processes based on the characteristics of the client the system to be built as well as the team.