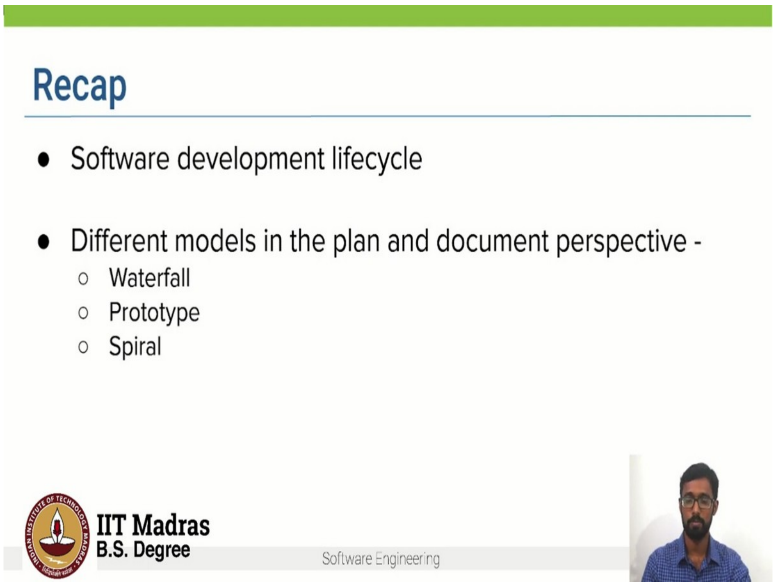
Software Engineering

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

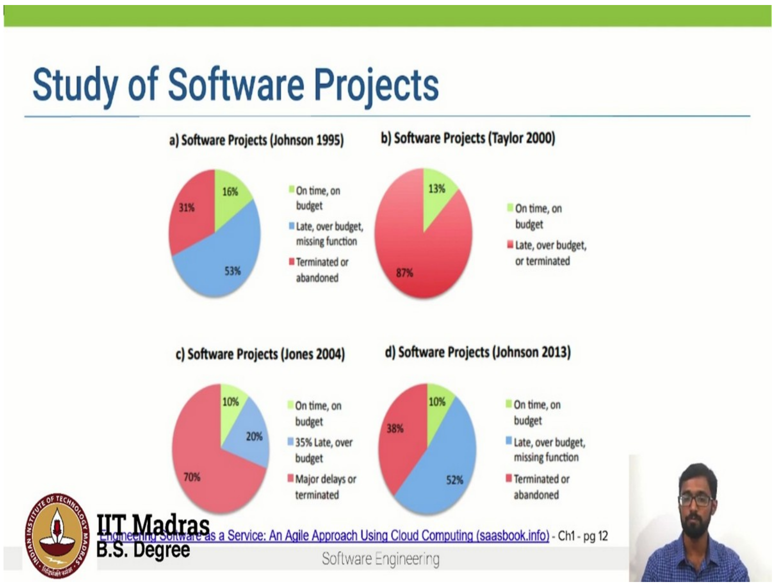
- Software development lifecycle
- Different models in the plan and document perspective -
  - Waterfall
  - Prototype
  - Spiral

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)



Study of Software Projects

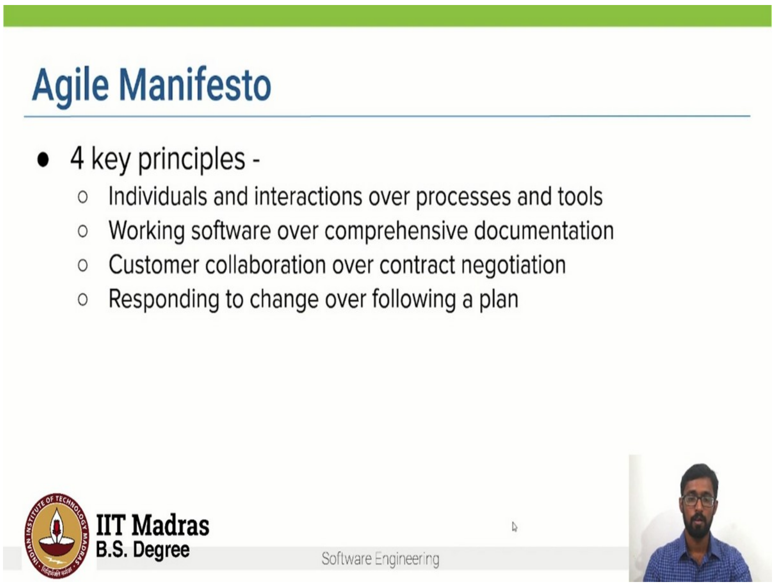
Engineering Software as a Service: An Agile Approach Using Cloud Computing (saasbook.info) - Ch1 - pg 12

IIT Madras
B.S. Degree

Software Engineering

So, the set of images on the screen, it shows the status of software projects across different years from 9095 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

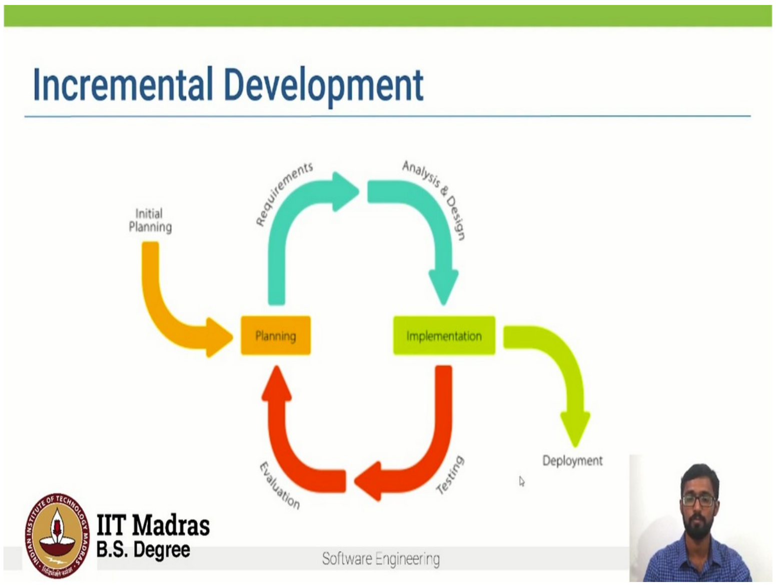
So, to address these problems or these difficulties, around 20 years ago, in February 2001, a group of software developers then met to develop a lighter weight software development lifecycle. And this came to be known as the Agile Manifesto. So, this manifesto the Agile

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 our 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.

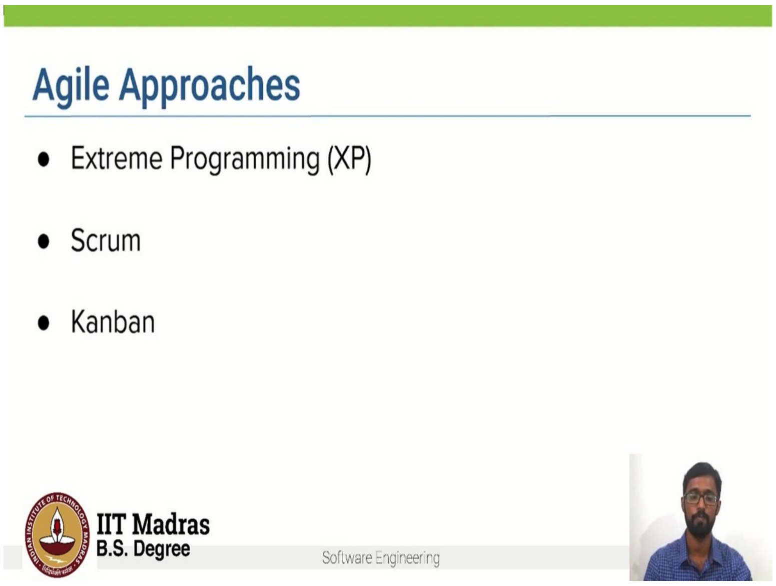
And finally, instead of following a strict and rigorous plan, agile emphasizes on responding to change whenever it is required. And what the manifesto states is that although the items on the right are valuable, the items on the left are more valuable in the Agile development process.

(Refer Slide Time: 4:25)


Incremental Development

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)



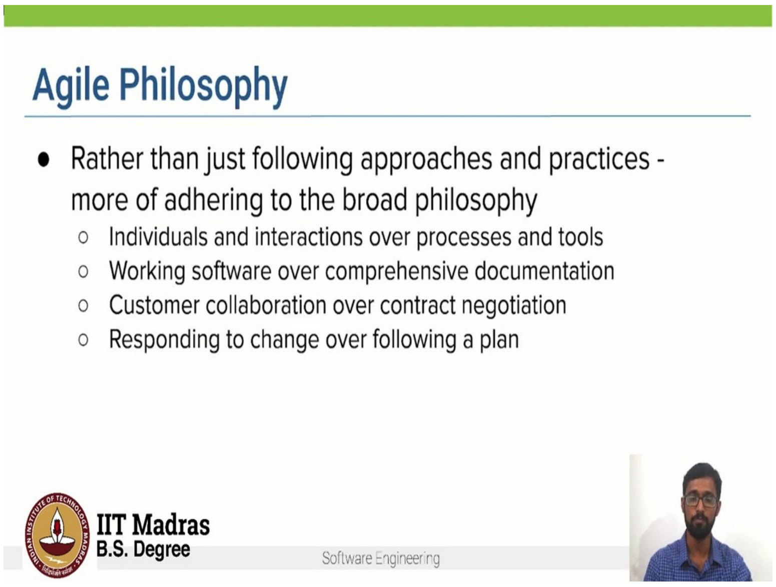
Agile Approaches

- Extreme Programming (XP)
- Scrum
- Kanban

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

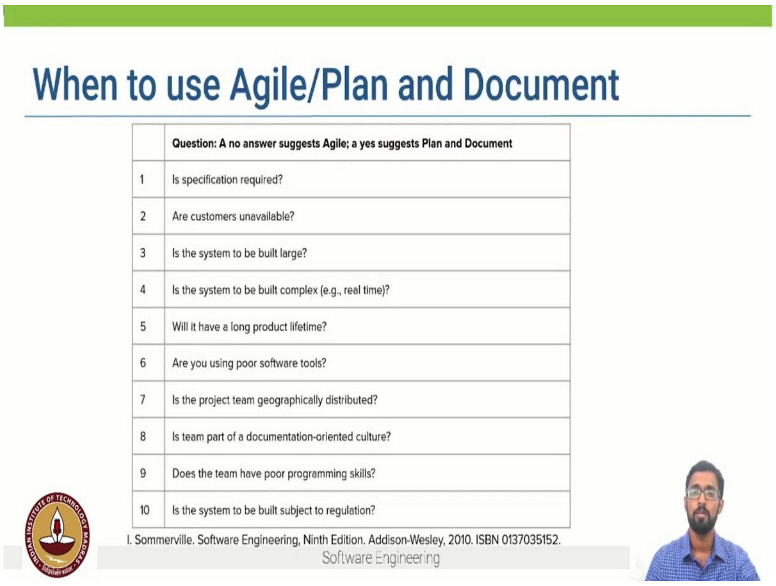
Plan and Document vs Agile - When to use?

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.

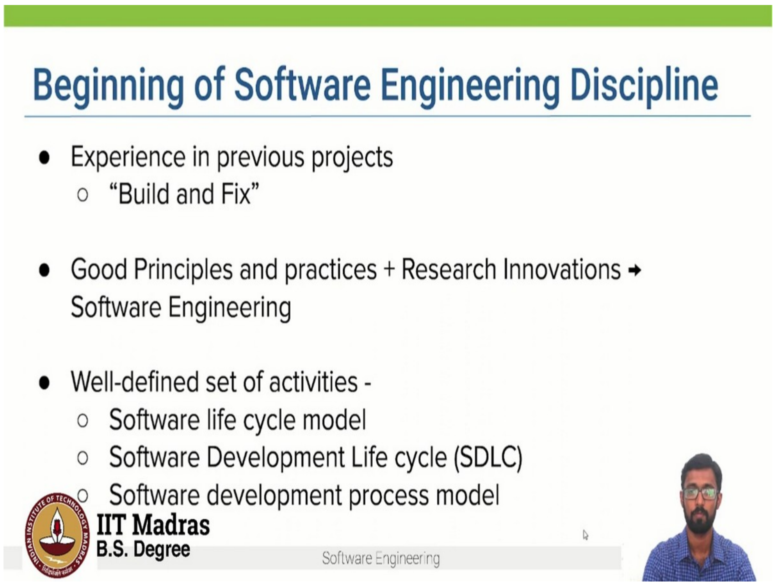
Another characteristics is about the team, your software team. So, if they are geographically distributed, if the team already is familiar with a documentation model, then it might be better to use a plan and document process and finally, is the system which we have to build is subject to several regulations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

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 - Plan and Document Perspective

(Refer Slide Time: 0:24)



Beginning of Software Engineering Discipline

- Experience in previous projects
  - "Build and Fix"
- Good Principles and practices + Research Innovations → Software Engineering
- Well-defined set of activities -
  - Software life cycle model
  - Software Development Life cycle (SDLC)
  - Software development process model

IIT Madras
B.S. Degree
Software Engineering

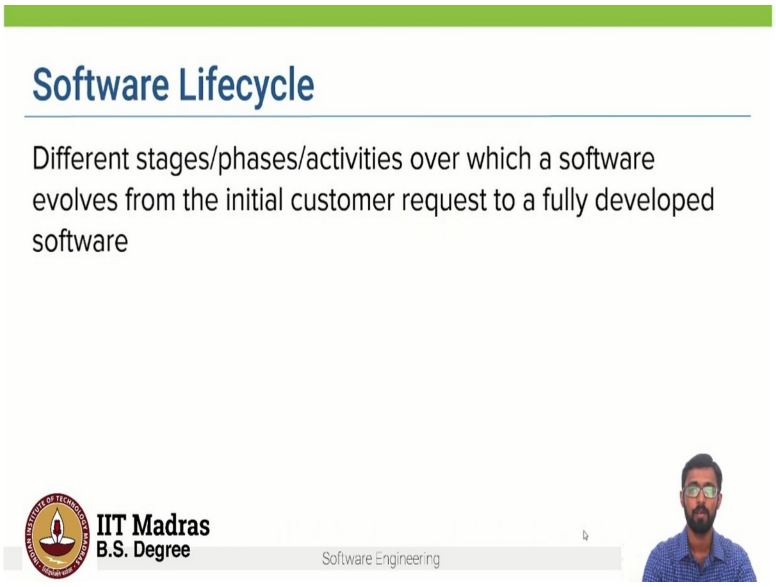
In the previous lectures of this week, we looked at various phases and activities in the software development process. We also looked at why these phases are important. So, learners many of you might have developed software projects before. So, let us reflect on how you went about developing software.

You might not have strictly followed the process of identifying requirements, then coming up with the design, then writing code testing it and so on. So, you might have followed an exploratory programming style. So, this is commonly known as the build and fix type of programming.

And here programmers, they typically start to write the program immediately after forming an informal understanding of the requirements. And once the program is complete, then they fix anything which does not match the users expectation. And this was how programming was done in the early 1950s and 60s. So, good programmers, they knew certain principles that helped them write good programs. And these principles could be things like thinking about the big picture view of the system, which is the design before you actually start writing the program.

And over time, these good principles and practices along with research innovations, they have been systematically organized into what is known as software engineering today. So, software Engineering emphasizes software development to a well defined and ordered set of activities. And these activities are commonly referred by various names such as the software lifecycle model, software development lifecycle or SDLC and the software development process model.

(Refer Slide Time: 2:36)



Software Lifecycle

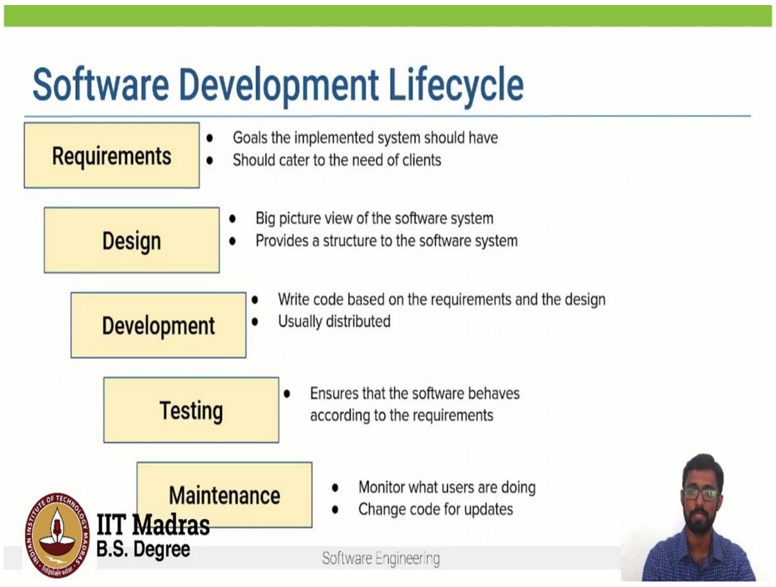
Different stages/phases/activities over which a software evolves from the initial customer request to a fully developed software

IIT Madras
B.S. Degree

Software Engineering

So, the software lifecycle is the different stages, the phases and activities over which is software evolves from the initial customer request to a fully developed software. And we saw some of these activities in the previous lecture.

(Refer Slide Time: 2:54)



Software Development Lifecycle

Requirements
- Goals the implemented system should have
- Should cater to the need of clients

Design
- Big picture view of the software system
- Provides a structure to the software system

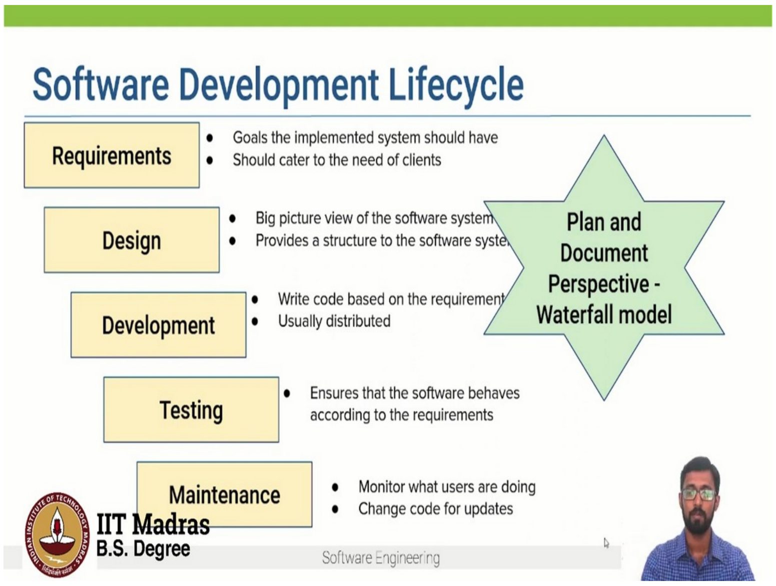
Development
- Write code based on the requirements and the design
- Usually distributed

Testing
- Ensures that the software behaves according to the requirements

Maintenance
- Monitor what users are doing
- Change code for updates

IIT Madras
B.S. Degree

Software Engineering



Software Development Lifecycle

Requirements
- Goals the implemented system should have
- Should cater to the need of clients

Design
- Big picture view of the software system
- Provides a structure to the software syste

Development
- Write code based on the requirement
- Usually distributed

Testing
- Ensures that the software behaves according to the requirements

Maintenance
- Monitor what users are doing
- Change code for updates

Plan and Document Perspective - Waterfall model

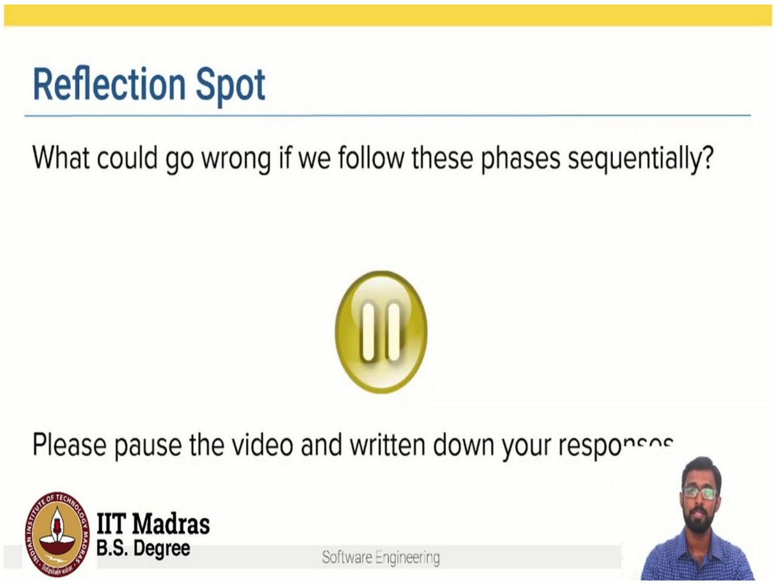
IIT Madras B.S. Degree

Software Engineering

So, we start the development by gathering requirements, which should cater to the need of the clients. Then we come up with a design which is a big picture view of the system. Then developers write code based on the requirements and the design. And then we do testing and maintenance. And this process is commonly known as the waterfall model, or the Plan and Document perspective.

It is known as waterfall model, because each of these phases occur one after the other. And it is known as a Plan and Document perspective because before we start writing code, we have to come up with a plan for the project, which includes extensive detailed documentation of all the phases of that plan. And progress is then measured against this plan.

(Refer Slide Time: 4:02)



Reflection Spot

What could go wrong if we follow these phases sequentially?

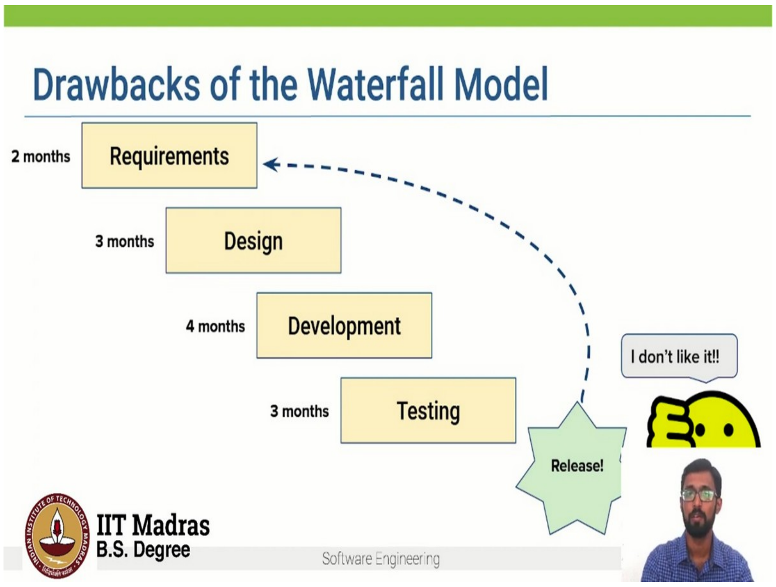
Please pause the video and written down your respon

IIT Madras B.S. Degree

Software Engineering

So, learners, let us reflect on this question. So, we saw that in the waterfall model, each of these phases follow one after the other. So, what could go wrong if we follow these phases sequentially? Please pause this video and think about some reasons before proceeding.

(Refer Slide Time: 4:35)



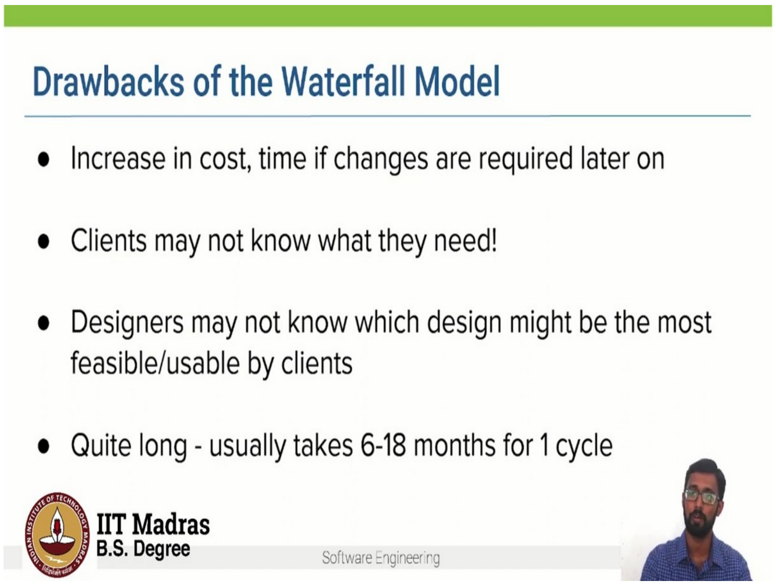
Drawbacks of the Waterfall Model

So, now let us look at some of the drawbacks of the waterfall model. Let us say, we are developing a software for a client and we talk to the client we gather requirements. Let us say that takes around 2 months. Then based on these requirements, we come up with the big picture view the design that takes around 3 months.

Then after that we start writing code for the design that takes around four months, and we test and fix bugs, which takes another three months. So, from the time of gathering requirements till the testing phase and finally the release, it takes around 12 months, it takes 1 year to develop the entire system and we release it to the client. But unfortunately, the client says that he or she does not like it.

There has been some misunderstanding regarding what the requirements are certain regarding certain aspects of the requirements or maybe the client changed his mind about certain aspects of some requirements, maybe they do not like the user interface. So, these are some drawbacks, and because of this, now, I have to go back and refine my requirements and start this process again. So, what are the drawbacks of the waterfall model?

(Refer Slide Time: 6:03)

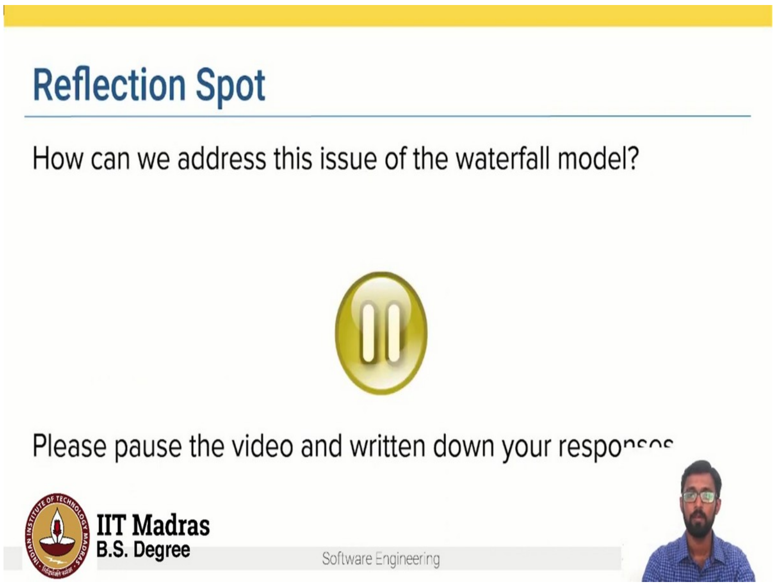

Drawbacks of the Waterfall Model

- Increase in cost, time if changes are required later on
- Clients may not know what they need!
- Designers may not know which design might be the most feasible/usable by clients
- Quite long - usually takes 6-18 months for 1 cycle

IIT Madras
B.S. Degree
Software Engineering

One as we saw, so, there is an increase in cost and time, if changes are required later on, as we saw in this scenario. Second, clients themselves may not know what they need. So, they might need to see something before they even realize what they actually need. And third, even the developers or the designers, they might not know which design might be the most feasible or most usable by the clients. And finally, as we saw one pace or one iteration can take maybe 6 to even 18 months for one cycle to occur. So, these are certain drawbacks of the waterfall model.

(Refer Slide Time: 6:58)


Reflection Spot

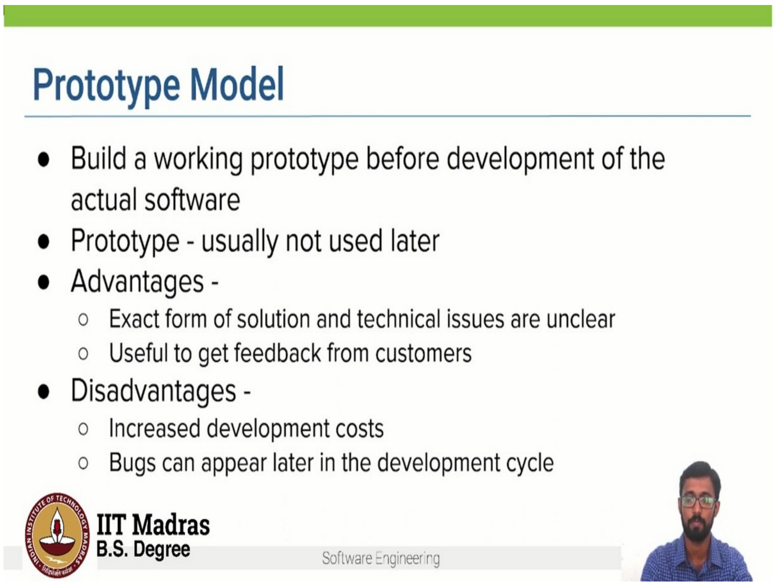
How can we address this issue of the waterfall model?

Please pause the video and written down your responses

IIT Madras
B.S. Degree
Software Engineering

So, now, as we have seen, there are certain drawbacks of the waterfall model, how can we address these issues or these disadvantages of the waterfall model? Can you think of some solutions? Please pause this video and think about some solutions before proceeding.

(Refer Slide Time: 7:28)



Prototype Model

- Build a working prototype before development of the actual software
- Prototype - usually not used later
- Advantages -
  - Exact form of solution and technical issues are unclear
  - Useful to get feedback from customers
- Disadvantages -
  - Increased development costs
  - Bugs can appear later in the development cycle

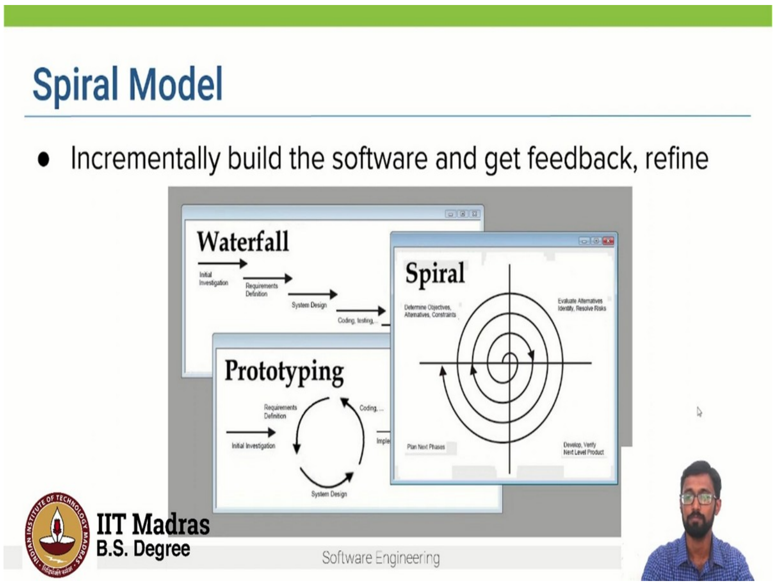
IIT Madras
B.S. Degree
Software Engineering

So, one solution can we get feedback from the customers earlier on. So, in this model, which is known as the prototype model, we build a working prototype of the system before development of the actual software. And this prototype has limited functional capabilities, and the performance might not be optimal compared to the actual software. The advantage is that it is easier for the customers to understand what they want, once they see this prototype as well as for developers and engineers to understand how to build it better once they have done it for the first time.

And usually the prototype it is not used later it is thrown away or new software is built. So, what are the advantages of the prototype model? So, one, it is useful when the exact form of the solution is not known and even the technical issues are unclear it is good to build a prototype and as we saw it is useful to get feedback from customers.

However, what is the disadvantage? One is that it can increase the cost of development for projects because as we saw the prototypes are not usually used later on and since we are not building the actual system, so what about risks and bugs which appear later in the software development in the development cycle. So, the prototype model cannot capture this.

(Refer Slide Time: 9:22)



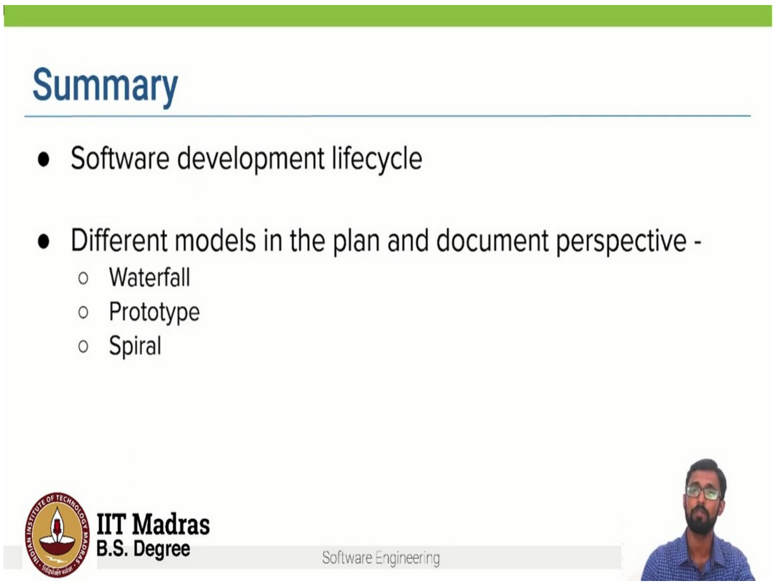
Spiral Model

- Incrementally build the software and get feedback, refine

Another solution can be to incrementally build the software, get feedback, refine and this module is known as the spiral model. So, the spiral model it combines both the waterfall and the prototyping module. And as you can see in this image, there are four phases. First, we determine the objectives, alternatives, constraints for this one iteration. And then we evaluate alternatives we resolved risks and then we develop and test and release the product to the client, and then we plan for the next phase. So, at each iteration, there was a refinement of the prototype compared to the previous version.

And unlike the waterfall model, where we document everything at the beginning, the requirement documents are developed across each iteration as and when they are needed. And the software eventually evolves into the final product. And we also involve customers at the end of every iteration and this helps in reducing misunderstandings between the client and the developers. But one drawback of this model is that the iterations still take long time so it takes around 6 to 24 months for each iteration.

(Refer Slide Time: 11:07)



Summary

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

IIT Madras
B.S. Degree
Software Engineering

So to summarize, in this video, we looked at the software development lifecycle, which is the different activities that need to be carried out for software development. We looked at different models in this plan and document perspective. We looked at the waterfall model where each of the phases are followed sequentially.

Then we looked at the prototype model where we build a prototype first and get feedback from users and the spiral model which combines the advantages of waterfall and the prototype model. Now, in the next video, we will be looking at the Agile perspective, which is a quite popular perspective nowadays to develop software. And we will be using many of the principles from the Agile perspective throughout the course.

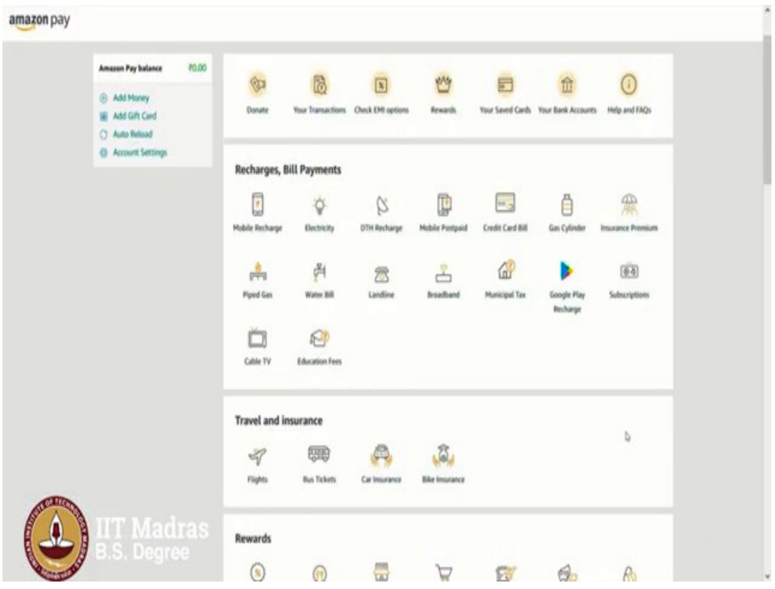
Software Engineering
Professor Sridhar Iyer
Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Professor Prajish Prasad
Computer Science, FLAME University

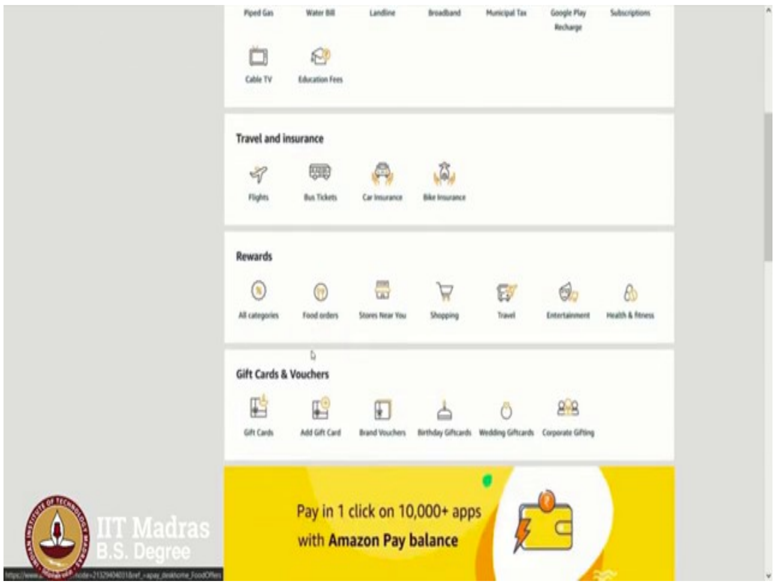
Software Development Process - Requirement Specification

Sridhar Iyer: In the previous video we spoke about the Amazon website and identified some components that it is made up of. The key idea we wanted to convey at that time was that a software system is made up of different, different components, which interact with each other to provide the functionality of the entire system.

Now it is also interesting to understand how each of these components are implemented. So for example, let us look at the Amazon Wallet, which is known as Amazon Pay, which stores your money and using which you do transactions. How do you think the Amazon Pay feature is implemented? Let us take a look.

(Refer Slide Time: 1:08)



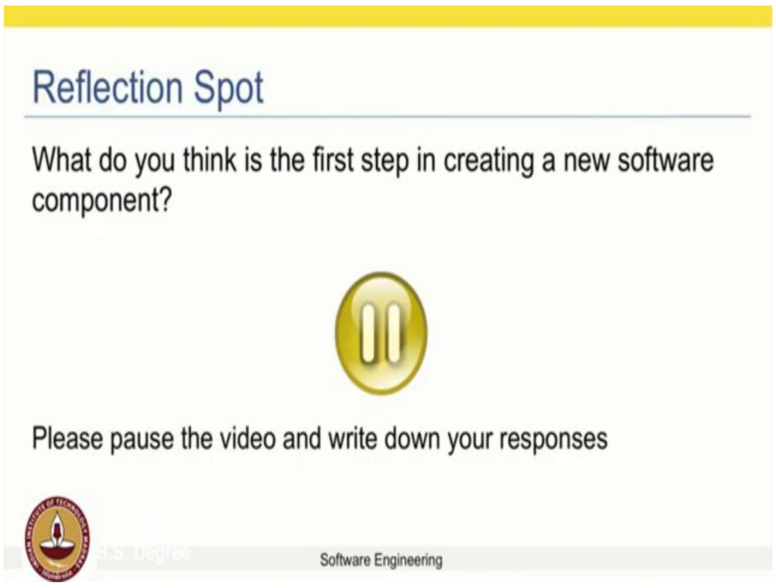


Prajish Prasad: Amazon Pay is a mobile wallet. A mobile wallet is a way to carry cash in digital format. You can link your credit card or debit card information as well as your bank accounts to a mobile wallet application or you can transfer money online to your mobile wallet.

Instead of using your debit or credit card to make purchases you can pay with your smartphone which has this mobile wallet. There are many categories like recharges, bill payments, travel and insurance, rewards, gift vouchers and so on. A mobile wallet has feature such as adding money and auto reload as well.

Sridhar Iyer: So, how do you think they started building the system, how did they come up with these functionalities?

(Refer Slide Time: 2:07)



Reflection Spot

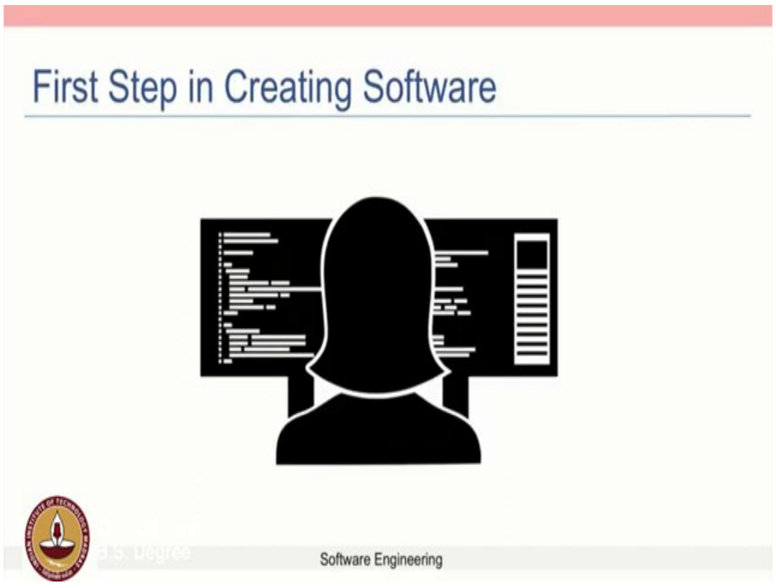
What do you think is the first step in creating a new software component?

Please pause the video and write down your responses

B.S. Degree
Software Engineering

What do you think is the first step in creating a software system? This is the key idea. What is the first step that we need to take? You can pause here, think about it for a moment and then proceed.

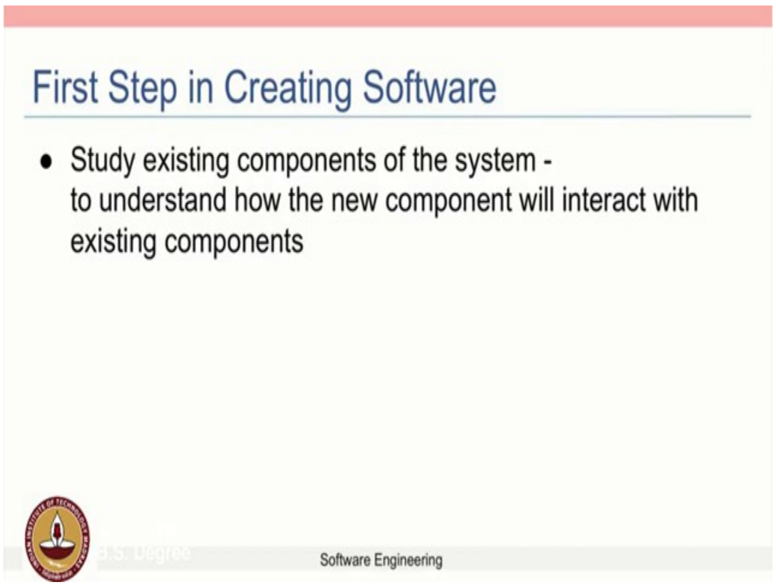
(Refer Slide Time: 2:31)



First Step in Creating Software

Prajish Prasad: So some of you would have thought that we first need to learn a programming language. Yes, that is in a way true. You need to write code for the software component using a programming language, but even before you start thinking of the solution you need to think of what problem you want to solve.

(Refer Slide Time: 2:52)



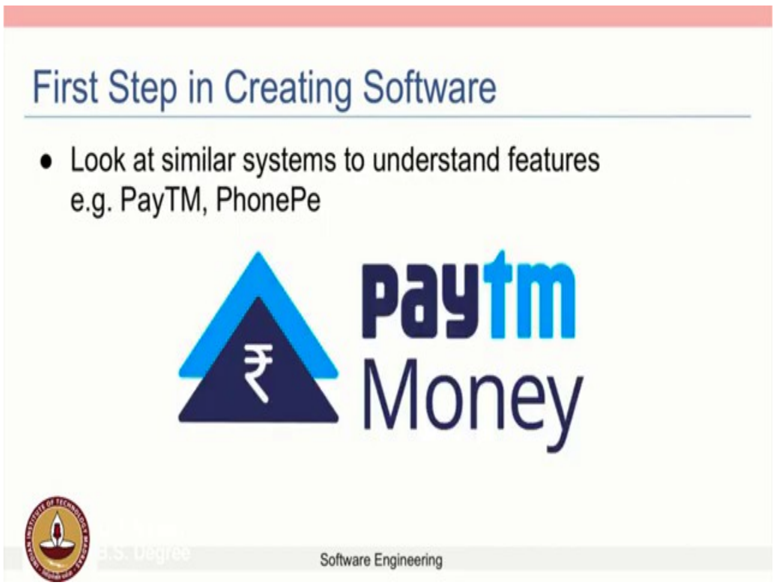
First Step in Creating Software

- Study existing components of the system - to understand how the new component will interact with existing components

Software Engineering

So another answer which you might have said is that we need to study existing components of the system. For example, other components of Amazon like the inventory management, the payment gateway and this can give us an idea of how this Amazon Pay feature will interact with these components.

(Refer Slide Time: 3:14)



First Step in Creating Software

- Look at similar systems to understand features e.g. PayTM, PhonePe

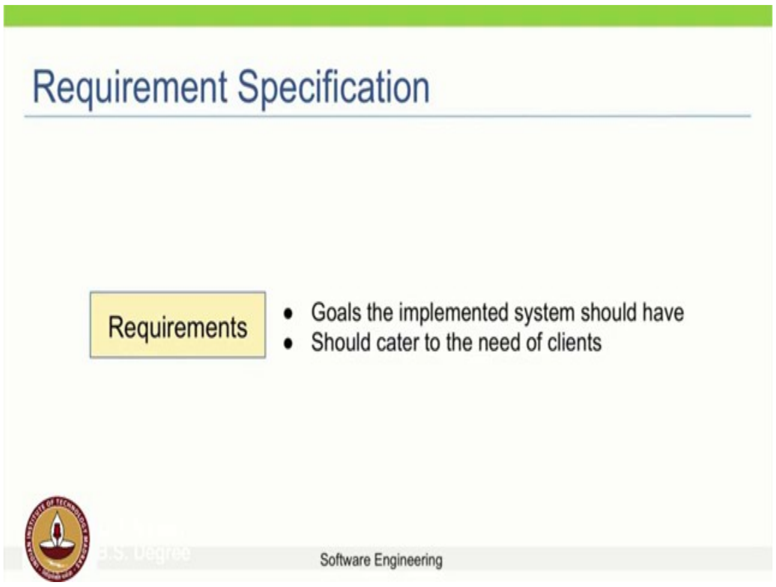
Paytm
Money

Software Engineering

Maybe I can also look at other wallet systems like Paytm, Phone Pay, etc. and see what features they have.

Sridhar Iyer: All these are valid answers. We need to first understand what is the problem we want to solve and based on an analysis of existing or similar systems we need to come up with an explicit set of goals for our own system or for what our implementation should provide.

(Refer Slide Time: 3:44)



Requirement Specification

Requirements

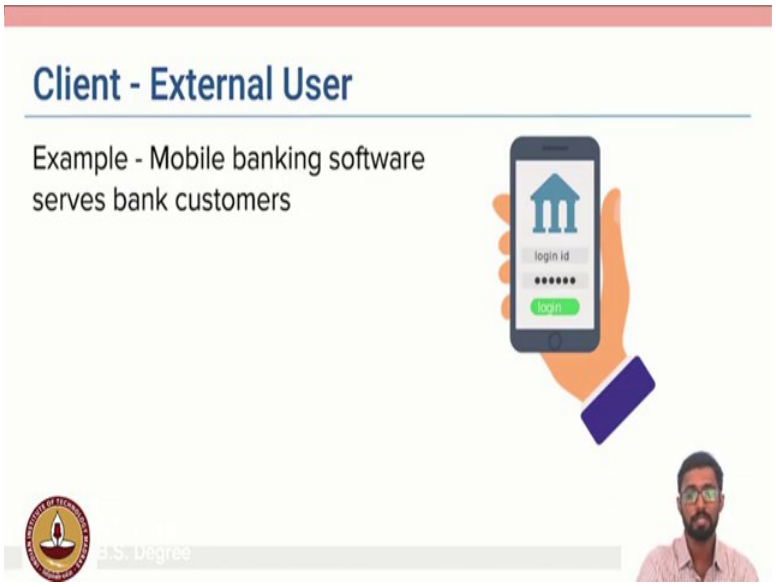
- Goals the implemented system should have
- Should cater to the need of clients

Software Engineering

Prajish Prasad: So these goals are called the requirements of the system. And this is usually the first step in the software development process. So ultimately the software component will be used by different types of users and these users are known as the clients. And we need to ensure that the requirements cater to their needs.

Sridhar Iyer: So who is a client? Does it mean that the client is the end user or does it mean the client is somebody who comes to buy something? That is the usual understanding of the term client. However, when we look at a software system, it is not always that the user is external to the system. Clients could be internal users also. For example, a client can be one component of the system, could be also a client of another component.

(Refer Slide Time: 4:47)

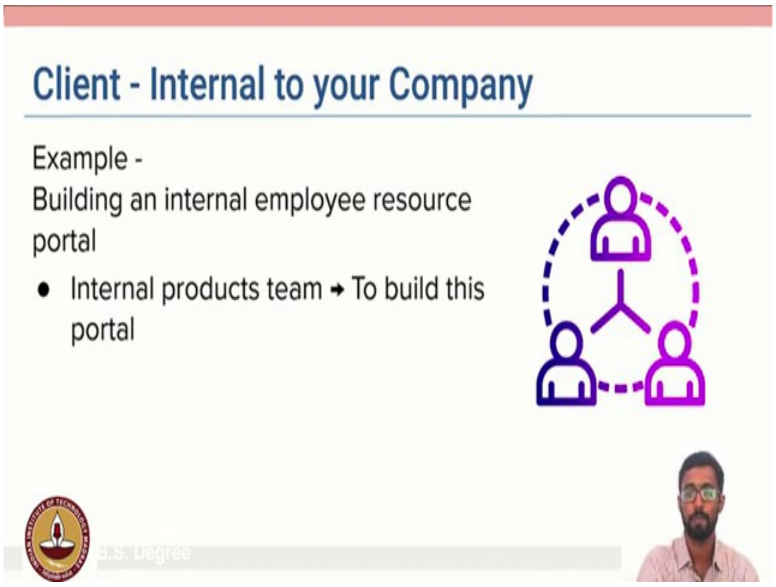


Client - External User

Example - Mobile banking software serves bank customers

Prajish Prasad: So client can be an external user and this is the most typical way to think about clients. For example, a mobile banking software serves customers of the bank by providing various banking features such as checking the account balance and transferring money.

(Refer Slide Time: 5:08)



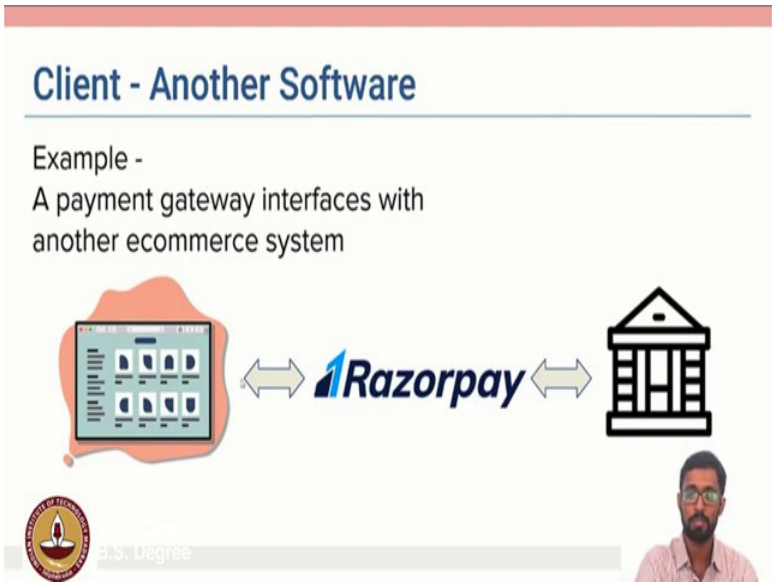
Client - Internal to your Company

Example -
Building an internal employee resource portal

- Internal products team → To build this portal

A client may be internal to your company as well. For example, a company might want to build an employee resources portal which contains information about various employees and teams in the company. To build such a system the company might form an internal products team, which communicates with other departments like Human Resources to understand the requirements.

(Refer Slide Time: 5:37)

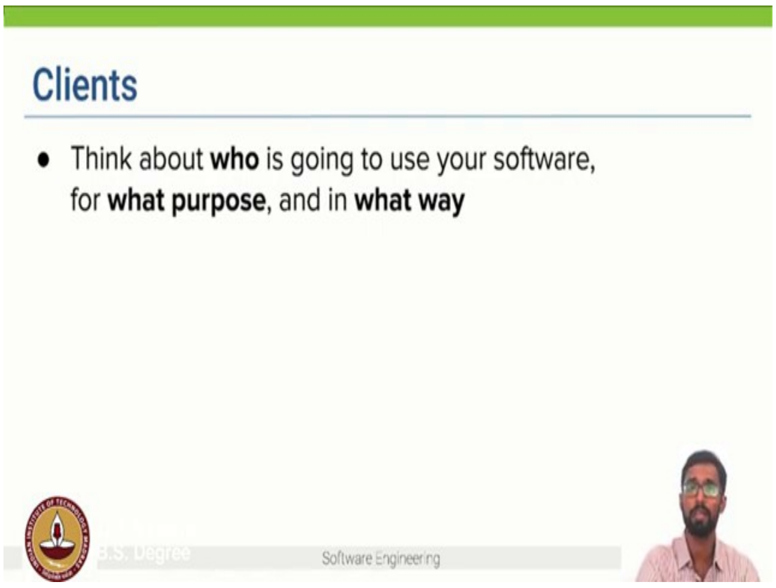


Client - Another Software

Example -
A payment gateway interfaces with another ecommerce system

Now the client can be another software. For example, a payment gateway like Razorpay interfaces with another ecommerce website or app to handle customer payments and refunds. Customer details are sent to the payment gateway by the ecommerce website and then the payment gateway communicates with the customer’s bank and sends the response back to the ecommerce website.

(Refer Slide Time: 6:09)



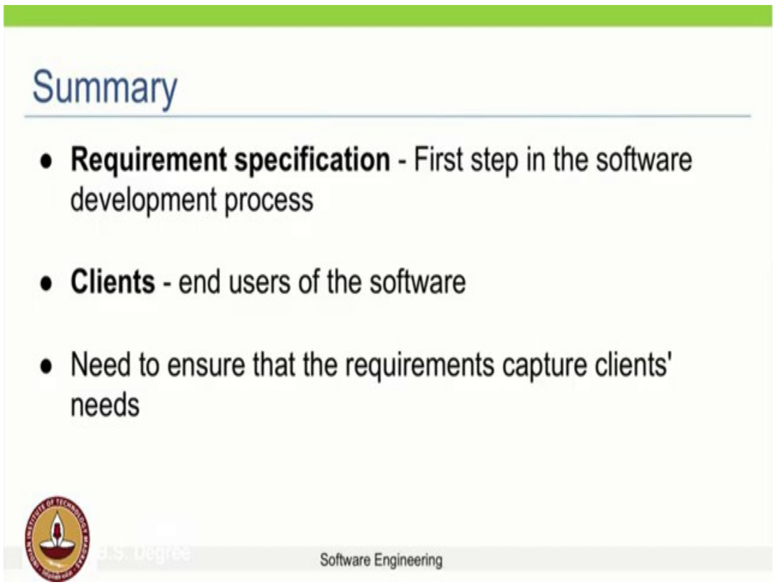
Clients

- Think about **who** is going to use your software, for **what purpose**, and in **what way**

So the important thing to note is that we need to think about who is going to use our software, for what purpose and in what way. This persona of the intended user must be alive in our mind as we think about creating software.

Sridhar Iyer: To summarize the first step in the software development process is to gather requirements.

(Refer Slide Time: 6:39)



Summary

- **Requirement specification** - First step in the software development process
- **Clients** - end users of the software
- Need to ensure that the requirements capture clients' needs

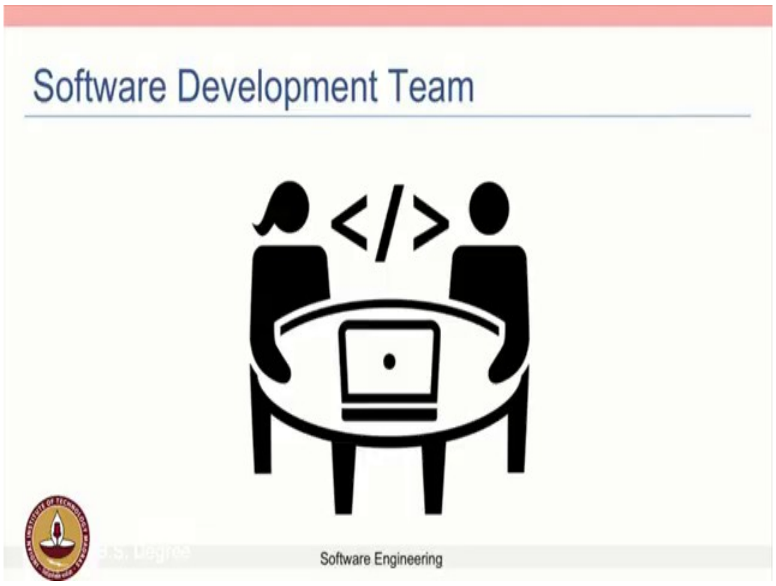
B.S. Degree Software Engineering

Why is this step important? Because this is the step that ensures that we have a good understanding of the users or the clients of the different components of our system. We need to ensure that these requirements are met at every stage, only then the entire software will meet the end users requirements.

Software Engineering
Professor Sridhar Iyer
Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Professor Prajish Prasad
Computer Science, FLAME University
Software Development Process - Software Design and Development

Sridhar Iyer: In the previous video we talked about the first steps of software development cycle; namely gathering of requirements. So what is the next step? Most people would think, okay, I am ready to start coding and they would jump to writing code. Is that really the next step? Let us see this with an example.

(Refer Slide Time: 0:48)

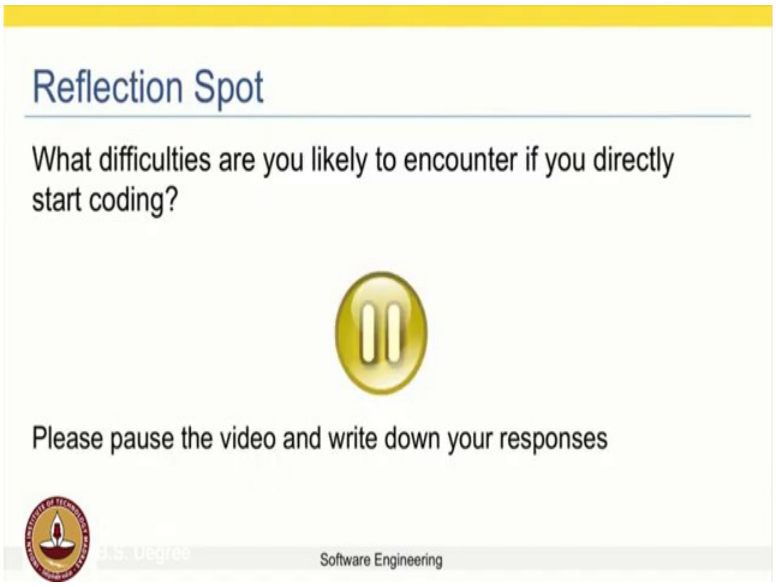


Software Development Team

Software Engineering

Imagine that you are a small team going to implement the Amazon Pay feature from the requirements. All of you are really excited to start implementing and directly jump into coding.

(Refer Slide Time: 1:02)



Reflection Spot

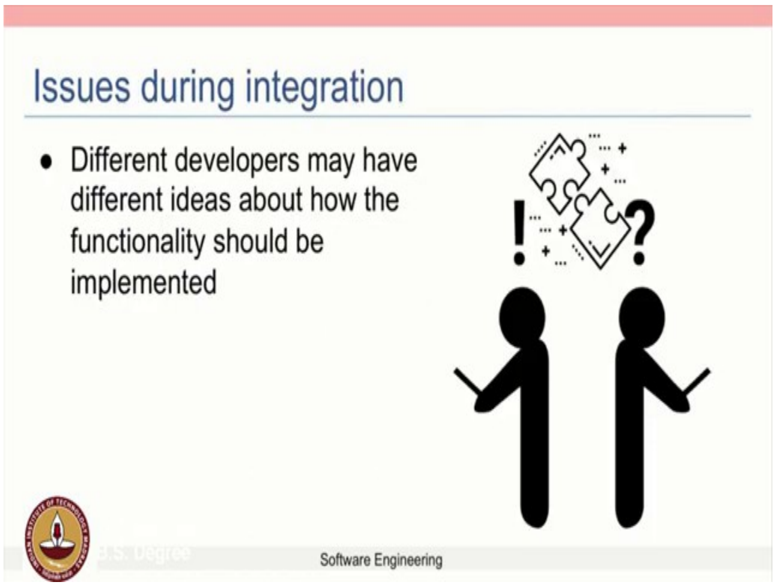
What difficulties are you likely to encounter if you directly start coding?

Please pause the video and write down your responses

Software Engineering

What difficulties are you likely to encounter? Pause here for a moment and think about it and write down one or two difficulties that may arise if we directly jump into coding after gathering the requirements.

(Refer Slide Time: 1:27)



Issues during integration

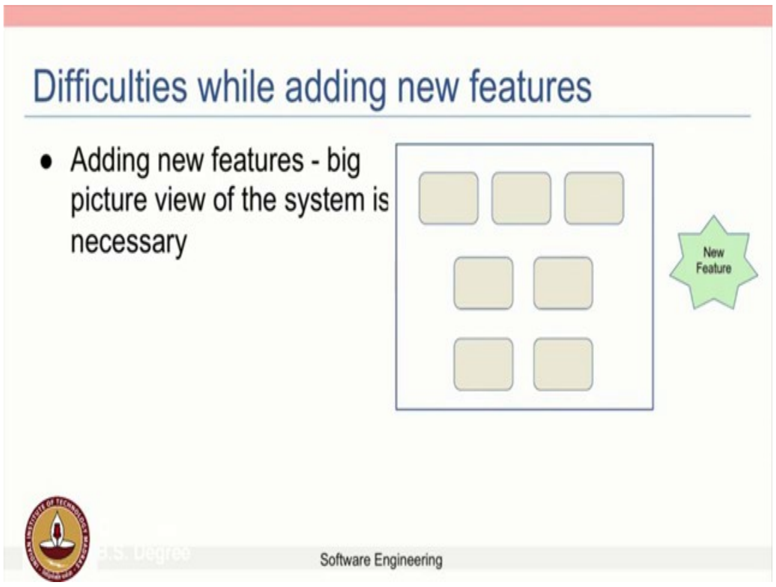
- Different developers may have different ideas about how the functionality should be implemented

Software Engineering

Prajish Prasad: So one issue can be that different developers they have different ideas about how the functionality should be implemented. So even if you have discussed about what should be done with your teammates you will most likely make changes in your code.

For example, you might add an argument to an existing function in your code and this can result in an error in the code of your colleague, which uses your function to develop her component. And this can be more problematic when different components created by different people have to interact with each other.

(Refer Slide Time: 2:10)



Difficulties while adding new features

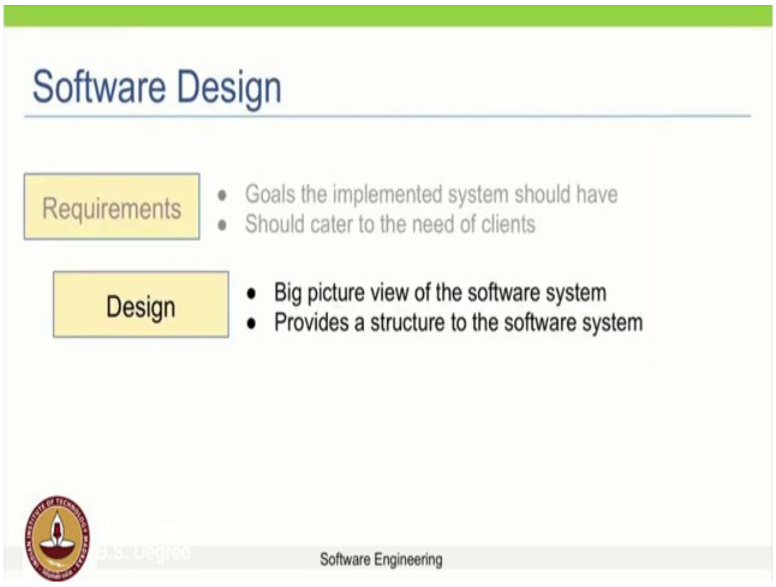
- Adding new features - big picture view of the system is necessary

New Feature

Software Engineering

And such a scenario can be a disaster. It will needlessly make the software complicated and the product will not be delivered on time. Also if I want to add new features later on it would help to have a big picture view of the system, so that I can decide how this new feature can be correctly integrated into the existing system.

(Refer Slide Time: 2:38)



Software Design

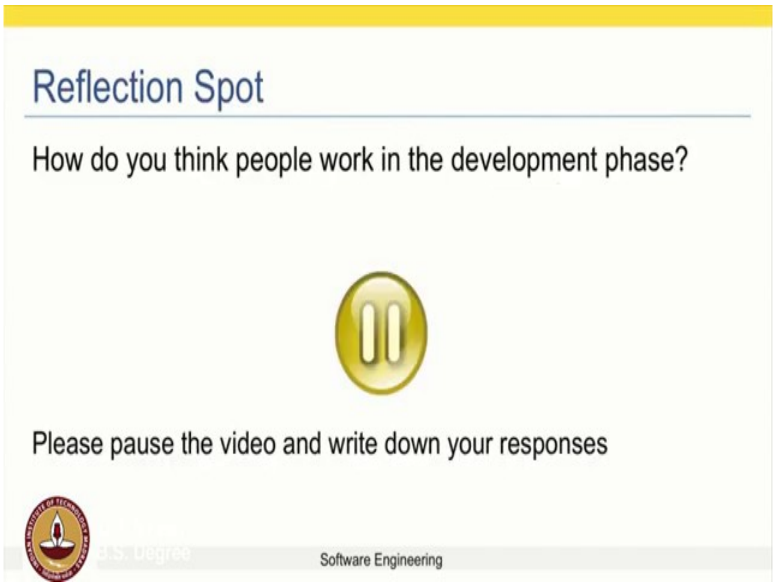
Requirements
- Goals the implemented system should have
- Should cater to the need of clients

Design
- Big picture view of the software system
- Provides a structure to the software system

Software Engineering

Sridhar Iyer: So this big picture view of the system is created in the design phase of the development cycle. We create this so that we know how to organize the code before we jump into implementing it and this provides the structure to the whole software system.

(Refer Slide Time: 2:58)



Reflection Spot

How do you think people work in the development phase?

Please pause the video and write down your responses

B.S. Degree
Software Engineering

Prajish Prasad: So after coming up with the design then we can go into implementing the system and write code. So learners here is a reflection spot for you. So what all comes to your mind when you see software developers writing code? How do you think people work in this development phase? You can pause this video and write down your responses before proceeding.

(Refer Slide Time: 3:37)

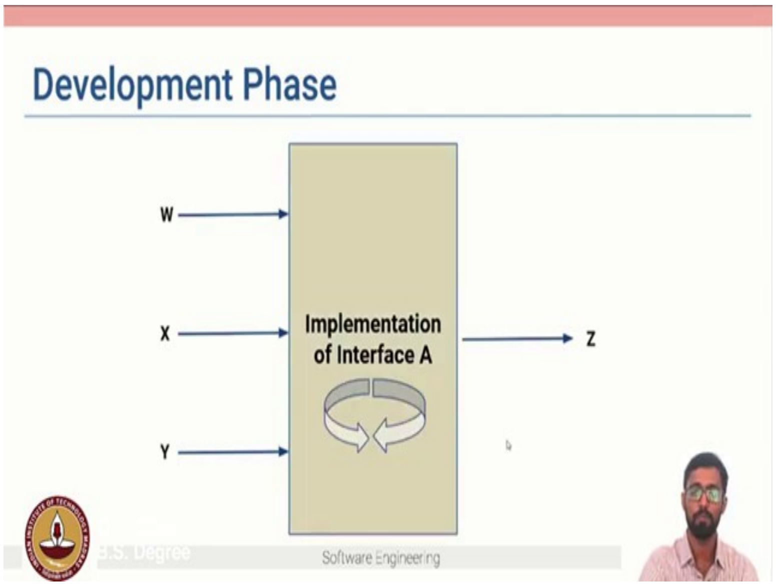
Development Phase

Development Phase

So when a feature is being implemented multiple developers work together and write code for the feature. They use tools like GitHub to collaborate and write code. And very often coding is done in a distributed manner with developers working in different locations and even in different time zones.

Hence, it is very important that everyone working on the code base has a consistent understanding about what the code does. And that is the reason why developers write documentation for their code and write precise interface definitions. So now what is an interface?

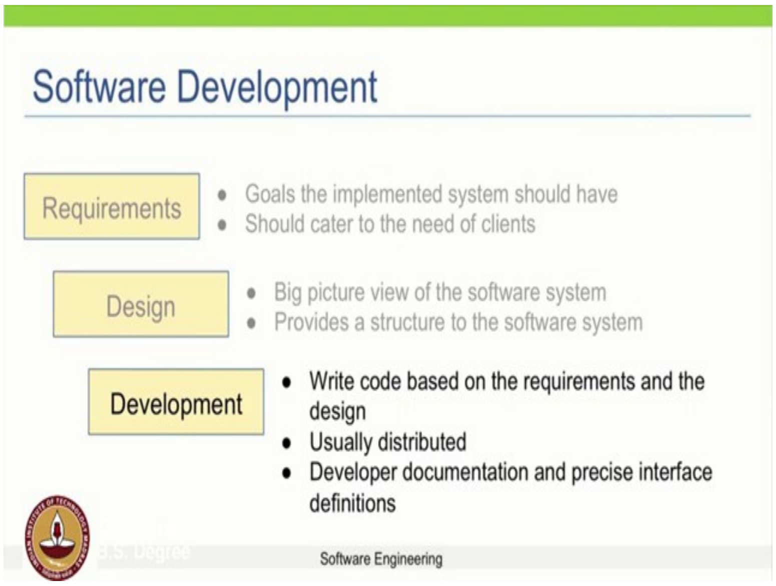
(Refer Slide Time: 4:27)


Development Phase

An interface is a description of the actions that your functions can do without describing the implementation in detail. The interface shows what requests are accepted and in what format the corresponding response is given that is the output. The code being implemented by the interface can change as long as the interface remains the same.

Sridhar Iyer: So in the development phase software developers write code based on the requirements and the design of the software system.

(Refer Slide Time: 5:11)



Software Development

Requirements
- Goals the implemented system should have
- Should cater to the need of clients

Design
- Big picture view of the software system
- Provides a structure to the software system

Development
- Write code based on the requirements and the design
- Usually distributed
- Developer documentation and precise interface definitions

B.S. Degree

Software Engineering

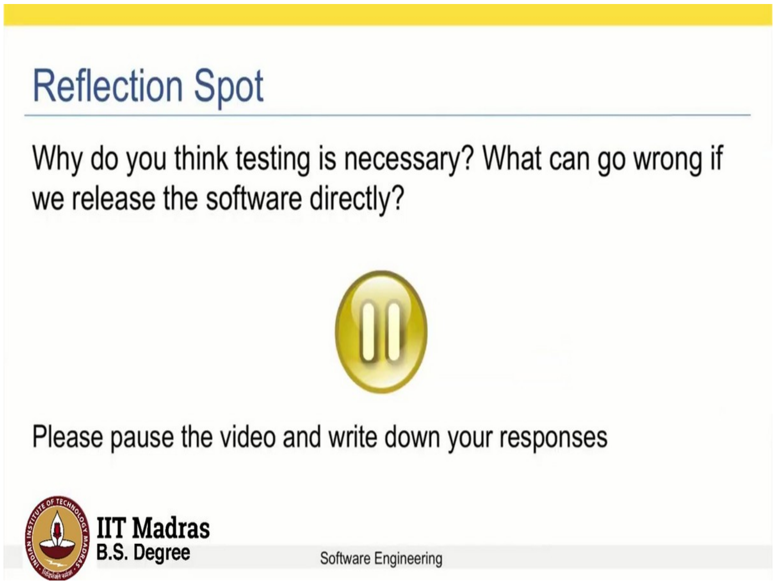
Development is usually distributed. These systems are too big for one person or one small team to develop. So multiple people write the code which implement the required functionalities, often multiple people are sitting in multiple time zones to do this. So to ensure now that we can put everything together and it functions everyone in the development team needs to write what are called interface definitions.

What does that mean? It basically means that what is the language that two components can talk. How do I expect a query? What is the form that I expect a query to take and what is the form in which I will provide the answer? So that is the simple way of thinking about interface definitions.

Software Engineering
Professor Doctor Sridhar Iyer
Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Doctor Prajish Prasad
FLAME University
Software Testing and Maintenance

Doctor Sridhar Iyer: In the previous video, we talked about the design and the development phases of software development. Does that mean that we are now ready to release our software? No, there are two more phases that we have to think about. One of them is testing, and the other one is maintenance.

(Refer Slide Time: 0:45)

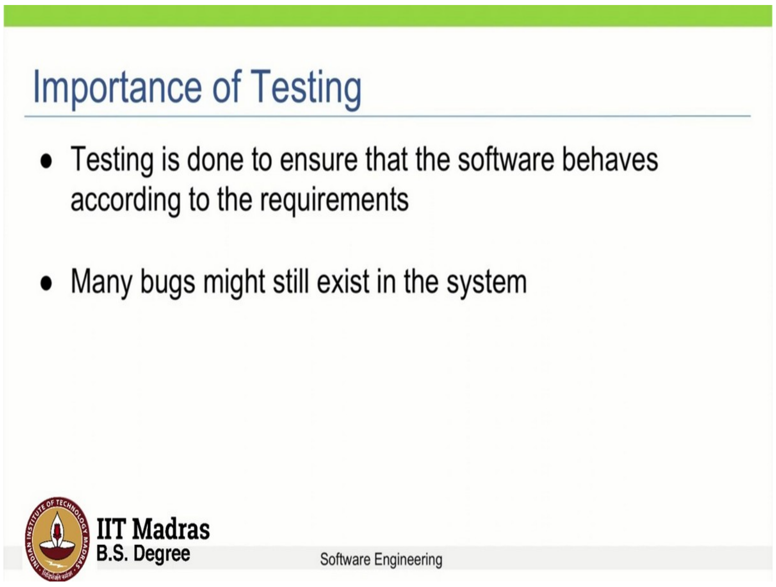


Reflection Spot

Why do you think testing is necessary? What can go wrong if we release the software directly?

Please pause the video and write down your responses

IIT Madras
B.S. Degree
Software Engineering



Importance of Testing

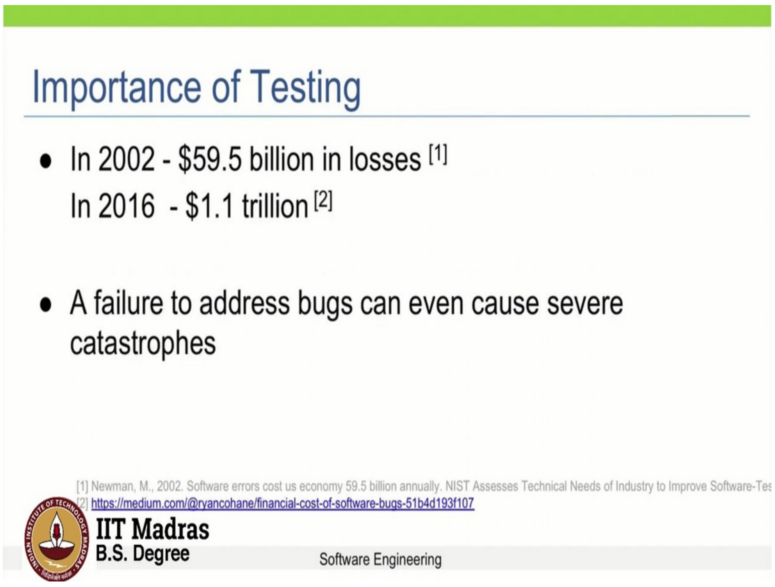
- Testing is done to ensure that the software behaves according to the requirements
- Many bugs might still exist in the system

IIT Madras
B.S. Degree
Software Engineering

So, here is a question for you. Why is testing so important? What will happen if we do not do rigorous testing of our software before we release it?

Doctor Prajish Prasad: So, after the software system is built, it is necessary that the software should behave according to the requirements. And at this stage testing of the system is done. And testing is important because many bugs and defects can still exist in the system.

(Refer Slide Time: 1:25)



Importance of Testing

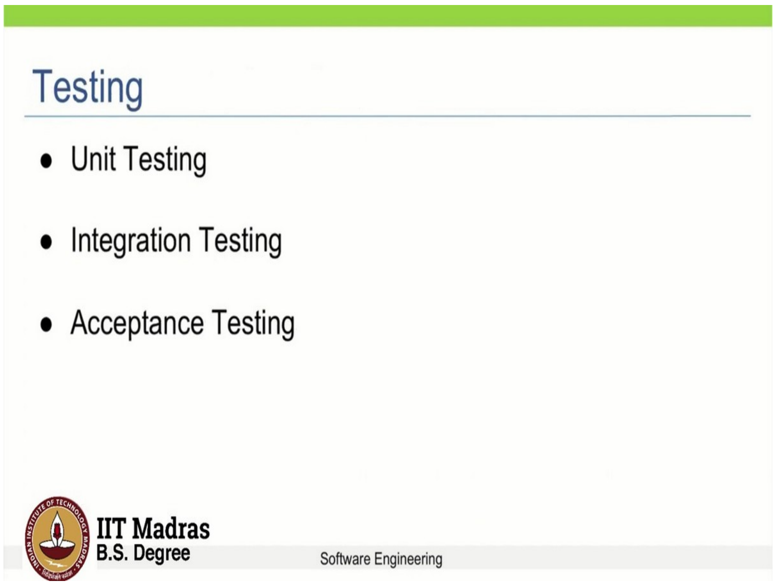
- In 2002 - $59.5 billion in losses [1]
  In 2016 - $1.1 trillion [2]
- A failure to address bugs can even cause severe catastrophes

[1] Newman, M., 2002. Software errors cost us economy 59.5 billion annually. NIST Assesses Technical Needs of Industry to Improve Software-Tes
[2] https://medium.com/@ryancohane/financial-cost-of-software-bugs-51b4d193f107

IIT Madras
B.S. Degree
Software Engineering

And these bugs can result in a loss of lot of money. For example, a study has shown that in 2002, software bugs cost the US economy around 60 billion in losses. And in 2016, this number jumped to 1.1 trillion. And the failure to address bugs can even cause severe catastrophes. And you can find more details about these in the additional resources which we have provided.

(Refer Slide Time: 2:02)



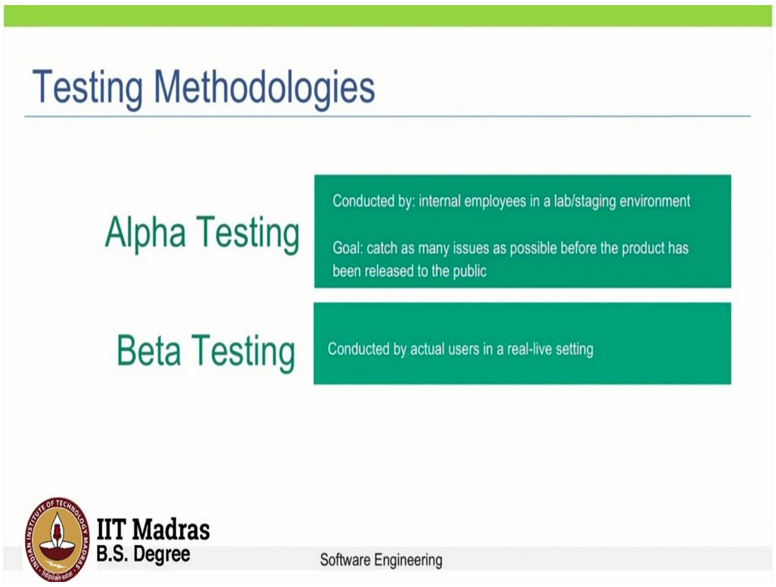
Testing

- Unit Testing
- Integration Testing
- Acceptance Testing

IIT Madras
B.S. Degree
Software Engineering

So, how is testing done? So, testing is done at different granularities. So, one type of testing is known as unit testing. So, unit tests, they focus on a single part of the whole application in total isolation. Usually, it is a single class or a single function, which is test. Integration testing is another type of testing, in which after now you have tested the individual parts, you now test how parts of the application work together as a whole. And finally, you have acceptance testing, where you check whether the requirements of the users are actually met by testing the application with end users.

(Refer Slide Time: 2:52)



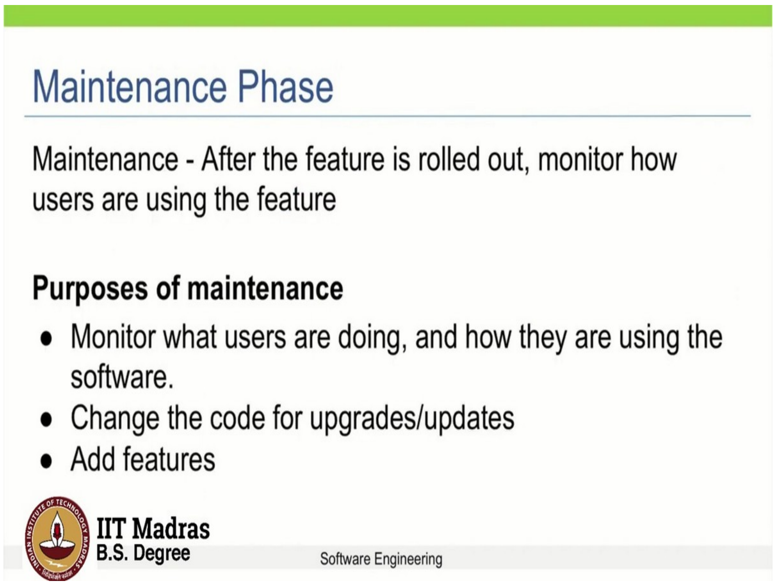
Testing Methodologies

Alpha Testing	Conducted by: internal employees in a lab/staging environment Goal: catch as many issues as possible before the product has been released to the public
Beta Testing	Conducted by actual users in a real-live setting

IIT Madras
B.S. Degree
Software Engineering

And there are two types of acceptance testing, alpha testing and beta testing. So, alpha testing is done by internal employees in a lab or a staging environment. And the goal of the alpha test is to catch as many issues as possible before the product has been released to the public. And then there is beta testing. Beta tests are conducted by actual users in a real live setting.

(Refer Slide Time: 3:22)



Maintenance Phase

Maintenance - After the feature is rolled out, monitor how users are using the feature

Purposes of maintenance

- Monitor what users are doing, and how they are using the software.
- Change the code for upgrades/updates
- Add features

IIT Madras
B.S. Degree

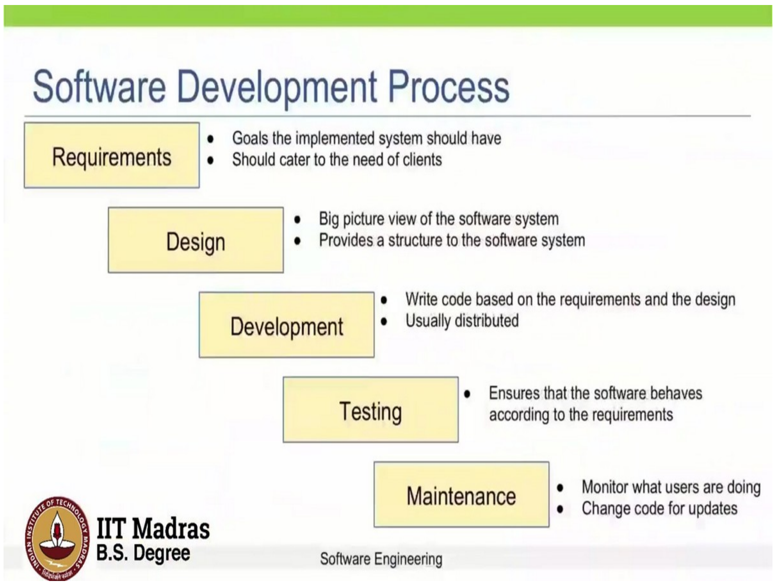
Software Engineering

Doctor Sridhar Iyer: And after adequate testing, the feature is rolled out. But it is necessary to monitor how users are actually using that feature. Are they using it the way we intend or is something else happening? We have to see if there are any difficulties or errors that they encounter. All of this is done during the maintenance phase.

So, the purpose of maintenance can be one to monitor what the users are doing and how they are using the software two change the code for upgrades, we are all familiar with this push of patch release. So, that is the maintenance phase of any software. And the third important thing is when we want to add features. So, we are all familiar with several software, where features have been added as we kept using them. Let us see this with an example.

Doctor Prajish Prasad: So, let us think about our Amazon pay example. So, even after releasing this feature, what difficulties or errors do you think users will face? There can be issues like for example, some conditions might have been missed, resulting in failures or maybe a simple UI issue in a very specific browser. So, the team identifies such issues in the maintenance phase and continues this process to fix such issues.

(Refer Slide Time: 4:58)



Software Development Process

Requirements
- Goals the implemented system should have
- Should cater to the need of clients

Design
- Big picture view of the software system
- Provides a structure to the software system

Development
- Write code based on the requirements and the design
- Usually distributed

Testing
- Ensures that the software behaves according to the requirements

Maintenance
- Monitor what users are doing
- Change code for updates

IIT Madras
B.S. Degree

Software Engineering

So, in the previous videos, we looked at different phases in the software development lifecycle. We looked at how we should gather requirements and how we should come up with the design, the development phase, the testing and the maintenance phase. So, each of the following weeks in this course, we will look at each of these stages in detail. We will look at the tools, the best practices in each of these stages. For example, what are the tools used to gather requirements? What are the practices for designing software in the development process and so on.

Software Engineering

Professor Sridhar Iyer
Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Professor Prajish Prasad
Computer Science, FLAME University

Thinking of Software in terms of Components

Sridhar Iyer: We are all familiar with buying things on an online platform like Amazon. We are familiar with ordering things online and we are not surprised when the goods land up at our doorstep. Behind it there is a huge system which actually makes this happen. So there is the vendor who is putting the things into the system, there is a logistics person who is trying to handle, ensure that the item reaches you in a timely manner and so on. Now how do you think this amazon system was built? Do you think it was built in one monolithic way?

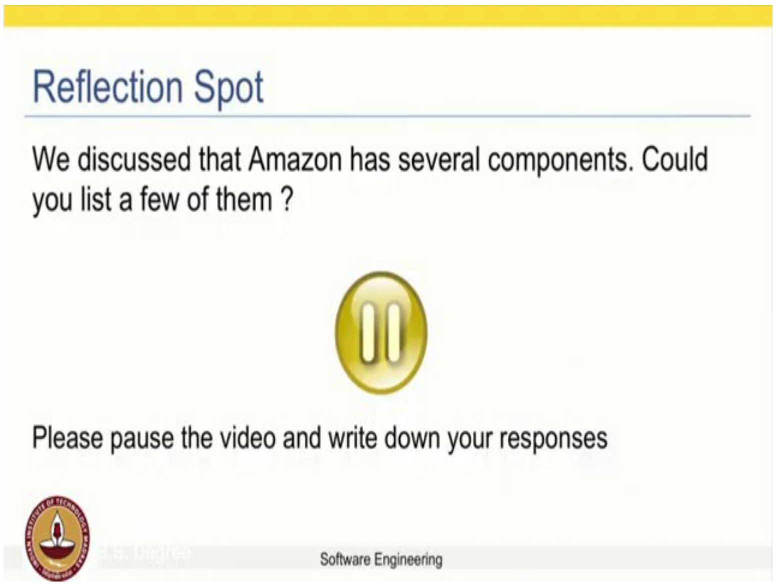
Prajish Prasad: So in most cases software is not built as a monolithic system, but it is built incrementally, one feature after the other. And moreover, often times we do not know at the start what all features are required to be implemented.

Sridhar Iyer: Yes, that is a good point. So we incrementally build certain features into the application and the one, the people who built Amazon would have also done the same thing. And even for building these features software teams have to ensure that they are divided into manageable components.

So components are a way of breaking the complexity of a system into manageable parts, so that different teams can work on different components of the system and put everything together in a timely manner. Also everyone need not know the workings of all the components. As long as I know that if I ask some response from a component and I know what sort of response it will give me that is sufficient for me to proceed with my work.

So the components can hide the complexity of the implementation and provide an interface to others who want to know what the component does.

(Refer Slide Time: 2:26)



Reflection Spot

We discussed that Amazon has several components. Could you list a few of them ?

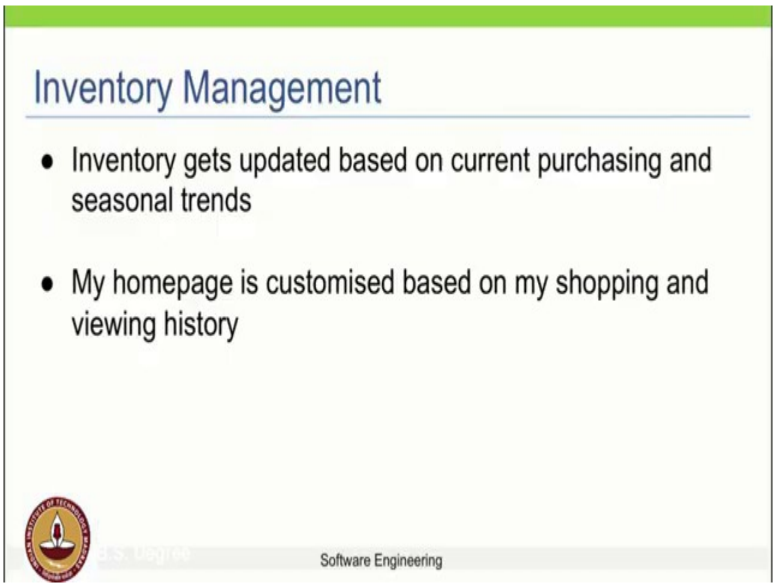
Please pause the video and write down your responses

Software Engineering

Now let us think about the components in the Amazon system. What do you think are some of the components? You can take a minute, pause here, think about what could the components in the Amazon system be if you were the, one of the developers of that system.

Prajish Prasad: So I think one major component in the Amazon system is, of course, their inventory management system. So they keep track of their inventory in such an intelligent way. So learners, you might be wondering what do we actually mean by intelligent, so let us think about what inventory management is. So inventory management is the act of measuring the amount, location, pricing, the mix of products available on Amazon.

(Refer Slide Time: 3:28)



Inventory Management

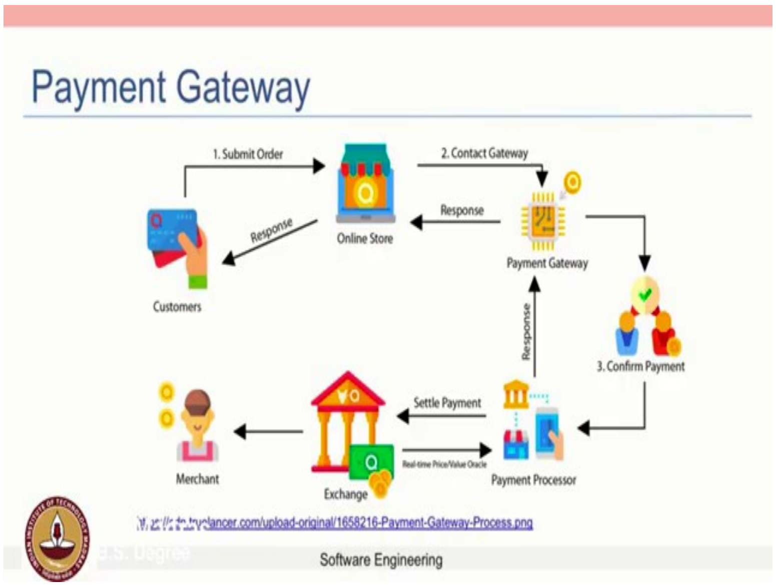
- Inventory gets updated based on current purchasing and seasonal trends
- My homepage is customised based on my shopping and viewing history

Software Engineering

So when we go to the Amazon homepage we see that our inventory gets updated based on the current purchasing and seasonal trends and fluctuating customer demand and logistics and analytics also play a large part in inventory management. So it is very interesting to see how my Amazon homepage could be very different from yours.

The reason being Amazon customizes my home page based on my shopping and viewing history. Another component can be their payment gateway. So the payment gateway is another component, which offers ease of payment for both buyer and the seller.

(Refer Slide Time: 4:20)



Payment Gateway

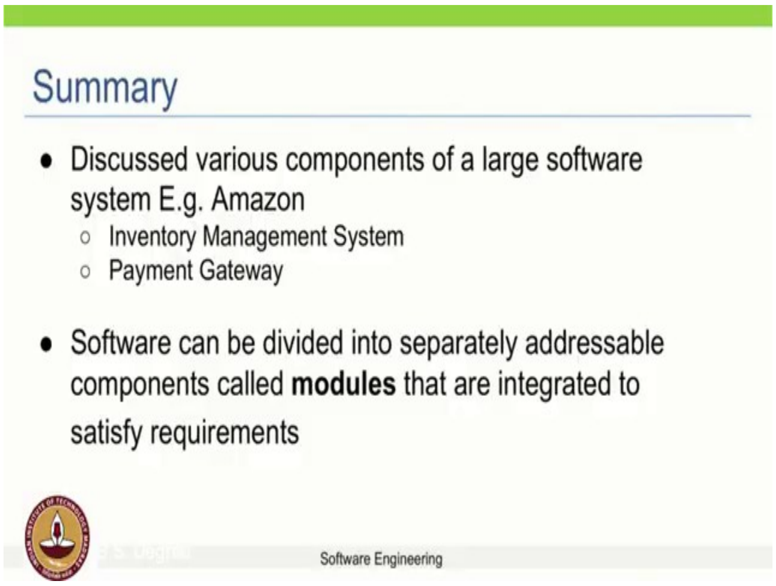
freelancer.com/upload-original/1658216-Payment-Gateway-Process.png

Software Engineering

So payment gateway is a service that authorizes the electronic payments like online banking, debit cards, cash cards, etc. So the payment gateway acts as a middleman between the bank and the merchant's website or application. When a user wishes to make a payment the Amazon website sends the encrypted card information to the payment gateway. Then the payment gateway confirms the validity of these details with the bank and the required amount of money gets transferred from the user's account to the Amazon account.

Sridhar Iyer: So we looked at two components of the Amazon system - the inventory management and the payment gateway. I am sure you can think of more components that makes up such a system. So in this video we looked at components of a large system such as Amazon and we looked a little more closely at two components, the inventory management and the payment gateway.

(Refer Slide Time: 5:26)



Summary

- Discussed various components of a large software system E.g. Amazon
  - Inventory Management System
  - Payment Gateway
- Software can be divided into separately addressable components called **modules** that are integrated to satisfy requirements

Software Engineering

The main thing that we have learnt is that nobody builds a system all at once for the entire functionality of the system. So a system, a large system is broken down into components and there are different teams that work on those different components. Each of these components now is called a module and they can be developed separately and integrated together later. Of course, there is a lot of complexity in doing so, because you need these modules to be able to talk to each other and so on and that we will see in the following videos.

Software Engineering
Professor Sridhar Iyer
Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Professor Prajish Prasad
Computer Science, FLAME University

Week 1 - Deconstructing the Software Development Process - Introduction

Sridhar Iyer: Welcome to this course on Software Engineering. We all know the importance of software today, but we need to know how to build such systems and how such systems are built. What do we have to do in terms of thinking about users? How do we do the coding? How do we do the testing and so on and so forth?

So this course is a primer to understanding how to design such software systems, how to do some of the implementation and some aspects of the testing and so on. So what are the expectations from you? As a learner, you are expected to put in at least six hours per week on the course, so that you can get the maximum out of it. And from our side we will ensure that there are activities and there are dialogues that happen throughout so that you do not feel that it is one long monolithic video that you are watching.

Prajish Prasad: So what is software engineering? So if you look at the history of software engineering, it started off as an art. So you had good programmers who wrote good code and you had bad programmers who did not write good code and over time the expertise of these good programmers, they were documented over time, which led to processes, models and strategies for effective software development.

And thus, the field of software engineering was born. So in this course, we will be looking at three important categories of topics. So one is the software processes. So what are different processes which are used in software development? You will be learning about processes like the waterfall model, the agile model. And what are common practices in these models, which are used in the industry?

A second category of topics is tools. So what are tools, which software teams use to build software? For example, what tools are used to capture requirements? How is software planning done? How is development done? How is testing done? So throughout the course you will be learning about different tools, which are used in the software development process. And third, we will also be talking about code. We will be talking about how you can organize your code, what are best practices for writing clean and effective code.

Sridhar Iyer: This course has been created using a learner centric MOOC model. What does that mean? It means that instead of simply having all the lectures in the beginning followed by all the assignments, followed by the exams, what we have tried to do is to interleave these in a way that is meaningful for you to maintain your engagement.

For example, there might be a video in which there might be a pause, where we ask you to think about something and maybe write down your thoughts, so this pause is very important for learning and that is called the reflection spot. Similarly, the videos might be short and immediately after the video we might want you to do some activity. So these are learning by doing activities, again which are very important for learning.

Then there will be discussion forums in which we will post questions to focus your discussion, so that you can have a focused discussion instead of a scattered one. And finally, there will be lot of resources, which will cater to your exploration in the direction of your interest. For example, some people may be more interested in coding, whereas some others may be more interested in how a software behaves actually in the field, whereas the third set of people may be more interested in talking to users of the software.

Prajish Prasad: Another key aspect of this course is the course project and through this course project you will be able to go through the entire software development lifecycle. So a Software Engineering course it can be very theoretical and the best way to get an understanding of what all you learn is to apply the processes, the practices, the tools in building an actual software.

So in each week you will have some intermediate milestones, which you need to complete to progress towards the goal of building a software product. So we hope you have a great time in this course and happy learning.