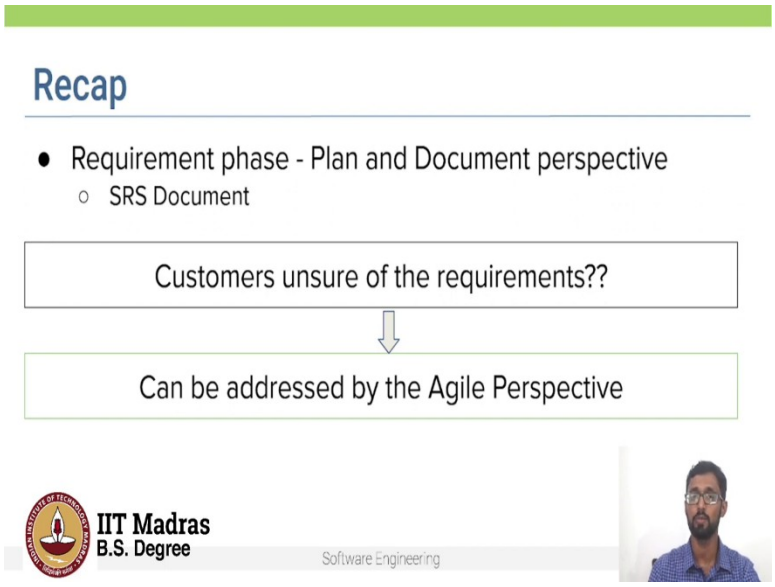
Software Engineering
Professor Sridhar Iyer
Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Professor Prajish Prasad
Department of Computer Science
FLAME University
Behaviour Driven Design – User Stories

(Refer Slide Time: 0:19)



In the previous videos, we looked at the requirement phase in the Plan and Document perspective, how a software team can gather and analyze requirements, and how they can organize this information in a document known as the SRS, or the software requirements

specification document. We saw that SRS documents serves as an agreement between the customers and developers.

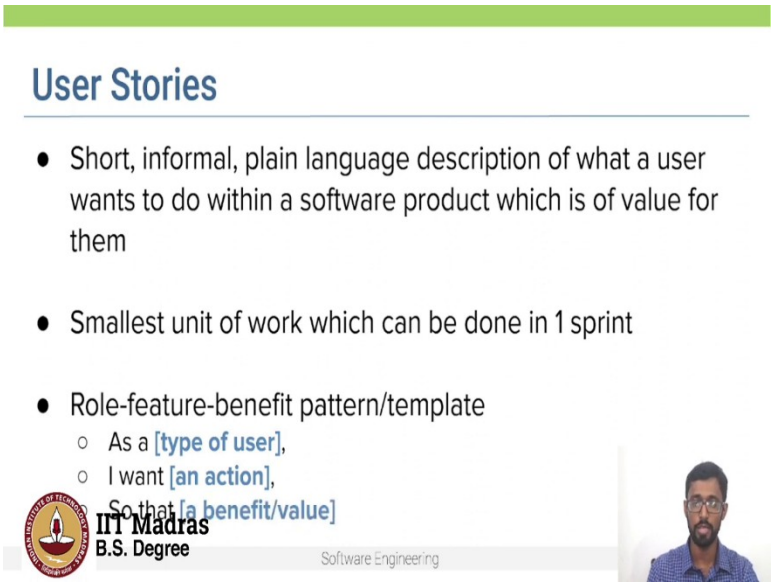
However, in many projects a customer themselves are unsure of the requirements and their understanding of their needs evolve and can change over time. So, how can we address this issue? The agile perspective can address some of these problems. So, in week 1, we saw that the Agile process involves working closely and interacting frequently with stakeholders to develop and refine requirements. It also involves building a working prototype and refining it by adding new features in sprints or iterations. And these iterations run typically every 2 weeks.

So, in the Agile perspective, we are basically in the maintenance mode as soon as we come up with some requirements, and then we start implementing features which match these requirements. So, this agile lifecycle begins with what is known as Behavior Driven Design.

(Refer Slide Time: 2:04)

So, let us look at what Behavior Driven Design is. Behavior Driven Design or BDD, it asks questions about the behavior of an application before as well as during development, so that stakeholders are less likely to miscommunicate. So, even in this case, requirements are written down. But these requirements are continuously refined to meet the stakeholder's requirements and expectations. The BDD version of requirements are known as user stories. And user stories take the place of what SRS was in the Plan and Document perspective.

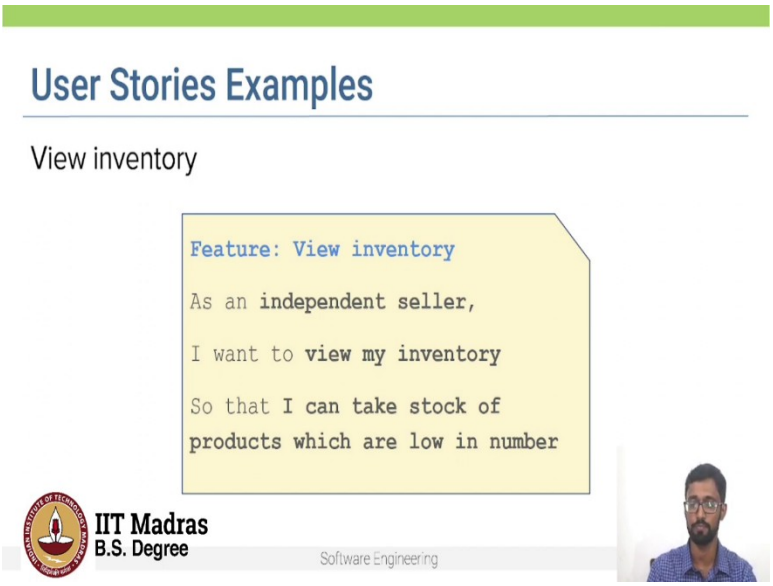
(Refer Slide Time: 02:56)



Now let us look at what user stories are. So, user stories are short, informal, plain language description of what a user wants to do within a software product, which is beneficial and of value for them. So, user stories come from the Human Computer Interaction community, or the HCI community. And these user stories are developed in 3 inch by 5 inch or 3 by 5 index cards.

So, we write these user stories in small carts. And user stories are the smallest unit of work, which can be done in 1 sprint. And 1 sprint is roughly 1 to 2 weeks. And user stories follow a particular pattern or template, which is known as the role-feature-benefit pattern, which states that as a particular type of user, I want a particular action, so that this that provides a particular value or benefit or the use. So, let us look at some examples.

(Refer Slide Time: 04:24)

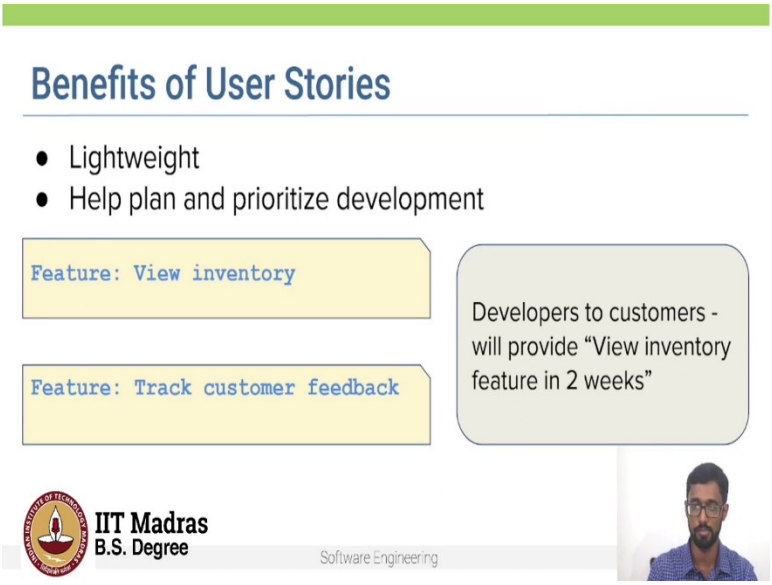


So let us take the example from the seller portal problem that we discussed in the previous lectures. Here is a requirement that we gathered from several stakeholders about viewing an inventory. So, user story for this is shown here. The feature is view inventory. So, as an independent seller, which is a type of user, I want to view my inventory, which is an action So, that I receive a particular benefit that is I can take stock of the products which are low in number.

(Refer Slide Time: 05:07)

Let us take another example of tracking customer feedback. So, what will the user story for this look like? So, here again, we follow the same pattern as an independent seller, which is the type of user, I want to view my customer's feedback for each product. And so that I can get a sense of the pertinent issues in my product. So, this is the value which we are providing to the user.

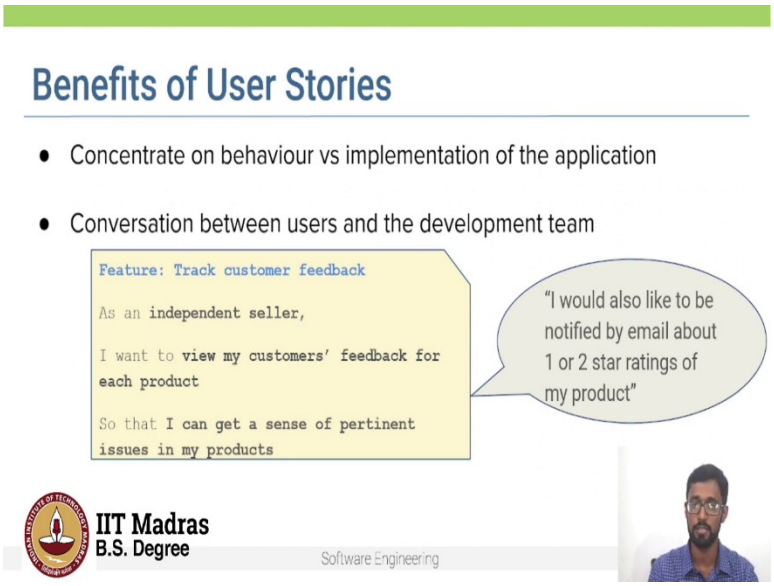
(Refer Slide Time: 05:46)



So, what are some benefits of these user stories? One, we saw that they are lightweight versions of the requirements, as opposed to the SRS documentation, which we saw earlier. Second, User Stories help stakeholders plan and prioritize development. So, for example, if I want a feature, which has to be implemented first, maybe for to get immediate feedback or to get a feel of the system, I can prioritize the development of the user story corresponding to that feature.

So, for example, we have two features of viewing inventory and tracking customer feedback. In this case, the client might want to implement the catalog and inventory first. So, the development team can tell the customers that we will provide the view inventory feature in 2 weeks.

(Refer Slide Time: 06:52)



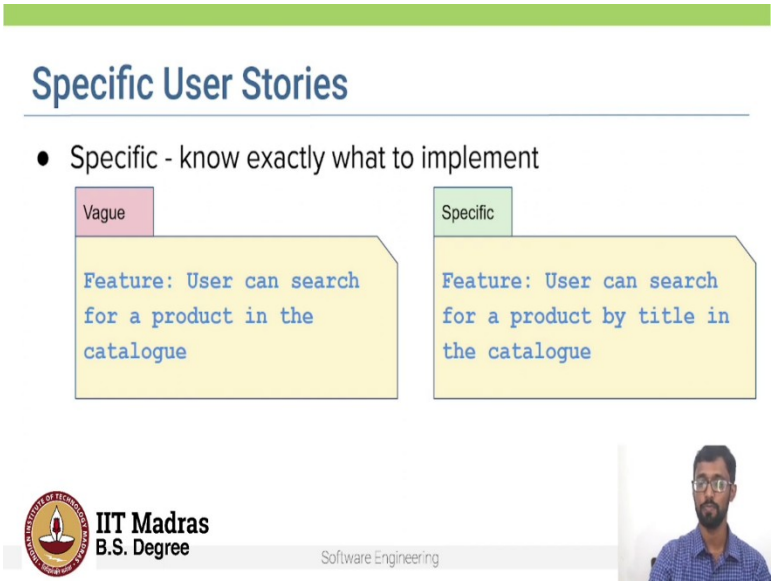
So, if you have noticed, we have not told the customers how we will implement these features. So, by concentrating on the behavior versus the implementation of the application, it is easier to reduce misunderstanding between stakeholders. Another benefit is that it facilitates conversations between the users and the development team. So, more than creating user stories, it is this conversation this back and forth between the different perspectives of each participant that will bring better simpler and more valuable solutions to user's problems.

For example, let us take this feature of tracking customer feedback. So, when we show this to a client, the customer or the client will realize that maybe I would also like to be notified by email about the products which have low ratings of 1 or 2 stars. And such modifications are common in the Agile perspective. And user stories can facilitate such modifications.

(Refer Slide Time: 08:13)

So, what are some guidelines to create good user stories? The acronym SMART can help us come up with good user stories. SMART user stories are Specific, they are Measurable, Achievable, Relevant, and Timeboxed. So, let us look at each of these in detail.

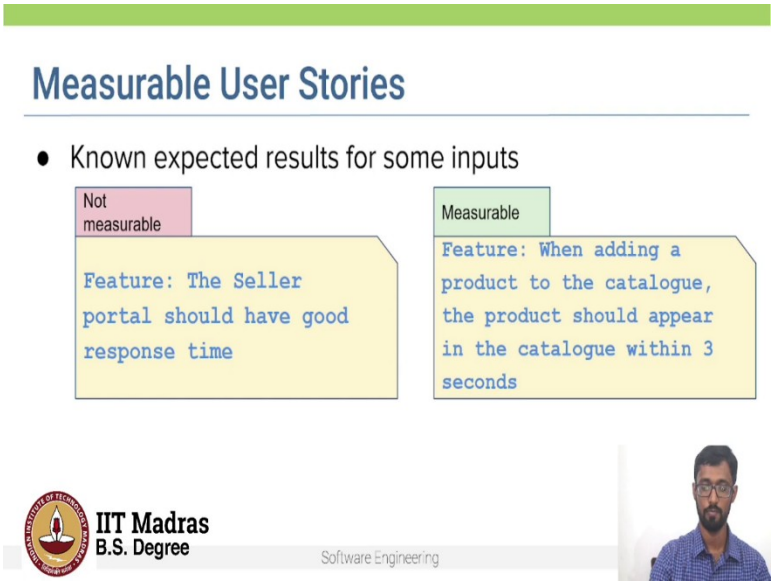
(Refer Slide Time: 08:43)



First, user stories should be Specific. So, specific user stories are essential so that we know exactly what to implement. So, let us consider this user story, which states that users can search for a product in the catalog. So, this is a bit vague, because we are not exactly specified. For example, what type of product it is, or based on what attribute can be searched

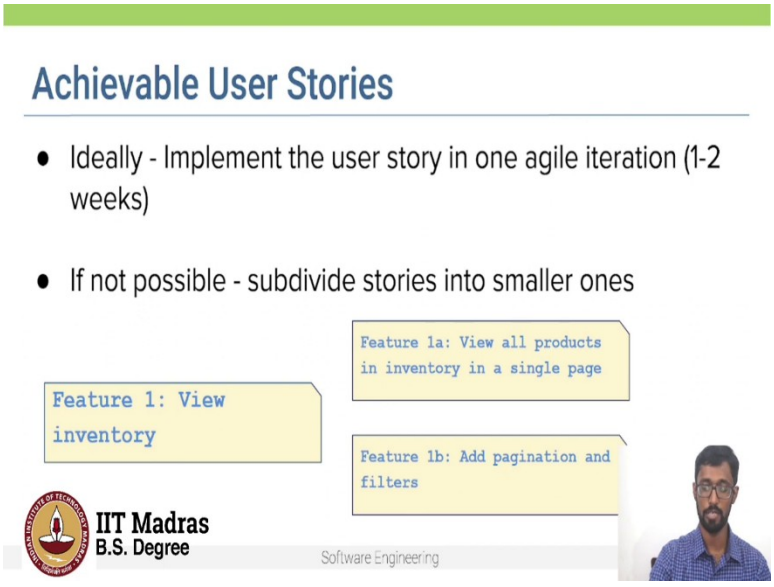
for a product. A more specific user story can be that the user can search for a product by title in the catalog.

(Refer Slide Time: 09:33)



Next user story should be Measurable. So, each story should be testable, which implies that there are known expected results for some good inputs. So, let us take this feature, the seller portal should have good response time. So, what do we mean by good? This is vague and it is not easily measurable, a more measurable user story can be, so when adding a product to the catalog, the product should appear in the catalog within 3 seconds. And this is measurable, and this is testable. So, make sure that your user stories are measurable.

(Refer Slide Time: 10:25)



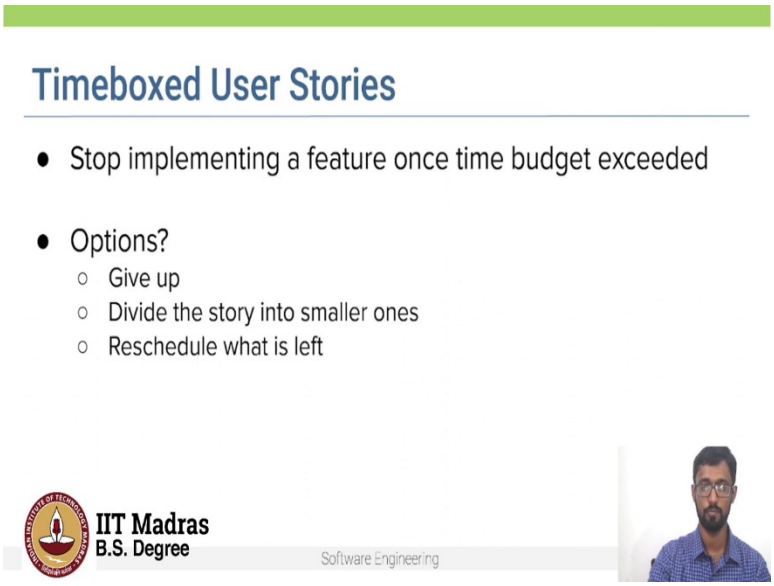
Next user stories should be Achievable. Ideally, we should be able to implement a user story in one agile iteration that is in 1 to 2 weeks. And if you are not able to complete one story per iteration, then they are too big. And you need to subdivide these stories into smaller ones.

So for example, view inventory might not be able, we might not be able to implement this in 1 week or in one iteration. Hence, we can divide it into two or three features that feature 1a is we implement the feature of viewing all products and the inventory in a single page. And then we add pagination and maybe filters can be the next feature. So, in this way, we make sure that one user story is completed in one agile iteration.

(Refer Slide Time: 11:38)

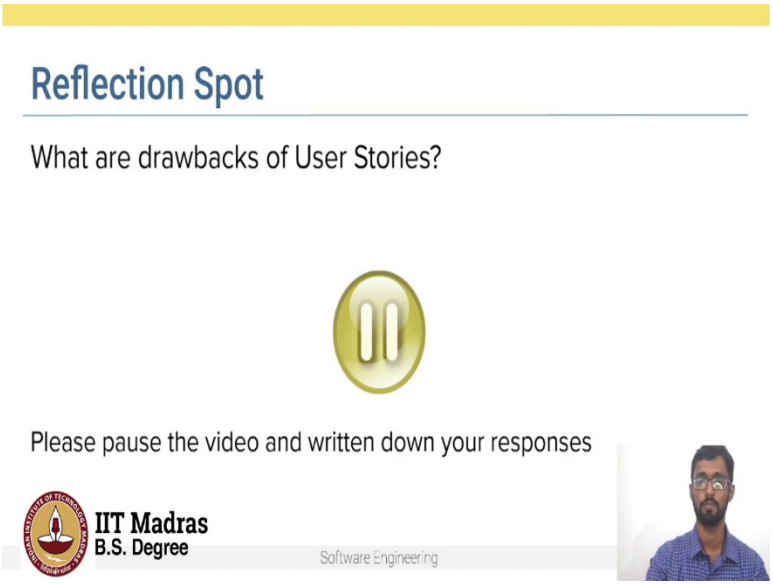
Next user stories should be Relevant. So, user story must have some business value to one or more stakeholders. So, how can we be sure that a user story is relevant? We should ask questions; we should ask questions like why is this feature needed? Is it really important is it adding any value? So, what if this feature is added? So, these questions can help us and ensure that the user stories which we add are relevant?

(Refer Slide Time: 12:18)



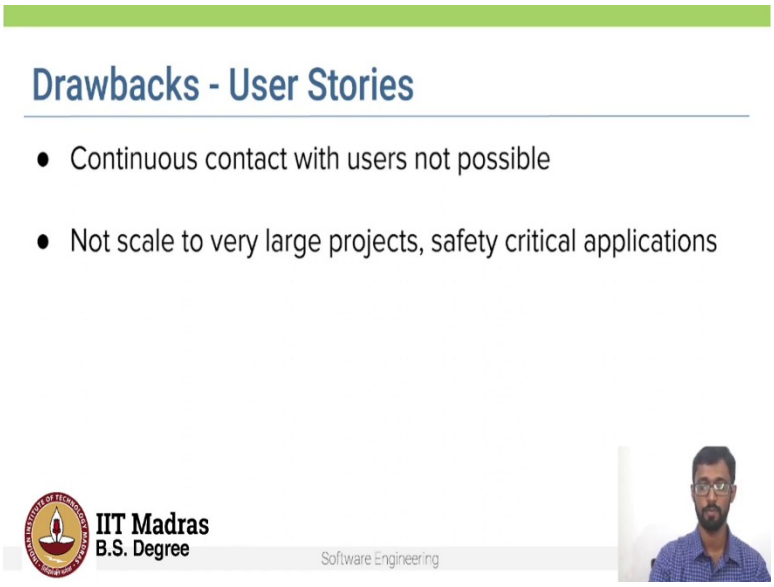
Next, user stories should be Timeboxed. So, Timeboxed means that we should stop implementing a feature once we have exceeded the time budget. So, what should we do if we have exceeded the time budget? Once we either give up? Or as we saw earlier, we divide the story into smaller ones. And or we reschedule what is left according to a new estimate, we revise our estimates. So, it is important to make sure that you are on schedule.

(Refer Slide Time: 13:10)



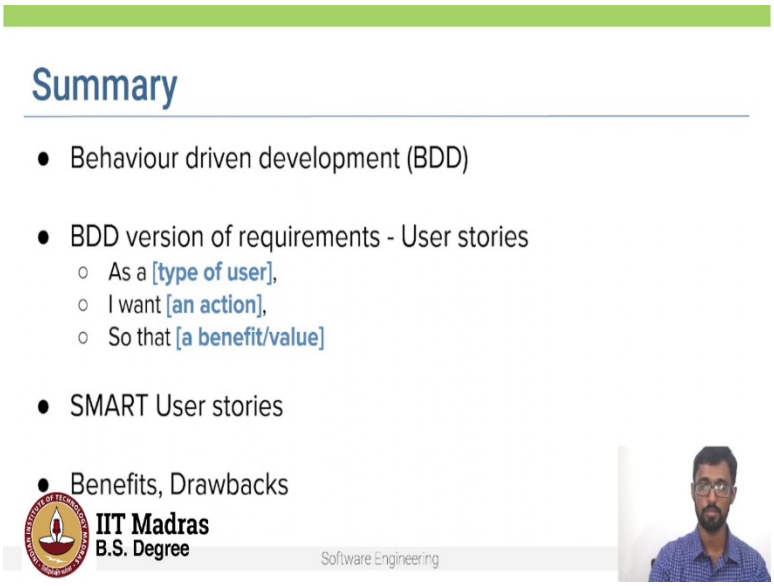
So we have seen the benefits of user stories, but let us reflect on this question for a moment on what are certain drawbacks of user stories? Are they always good? Can they not be used in certain situations? Please pause this video and write down your responses before proceeding.

(Refer Slide Time: 13:39)



So, what are certain drawbacks of user stories? One, it may be difficult or too expensive to have continuous contact with customers throughout the development process. Because customers may not want to participate or they may or may not be available at all times. Second, this approach may also not scale to very large software development projects or even safety critical applications because in such projects, extensive planning and documentation are required before the actual implementation, and especially for safety critical applications, there is no room for errors or refinement. So, the key point to remember is that software development methodologies depend on the type of application you are building.

(Refer Slide Time: 14:43)

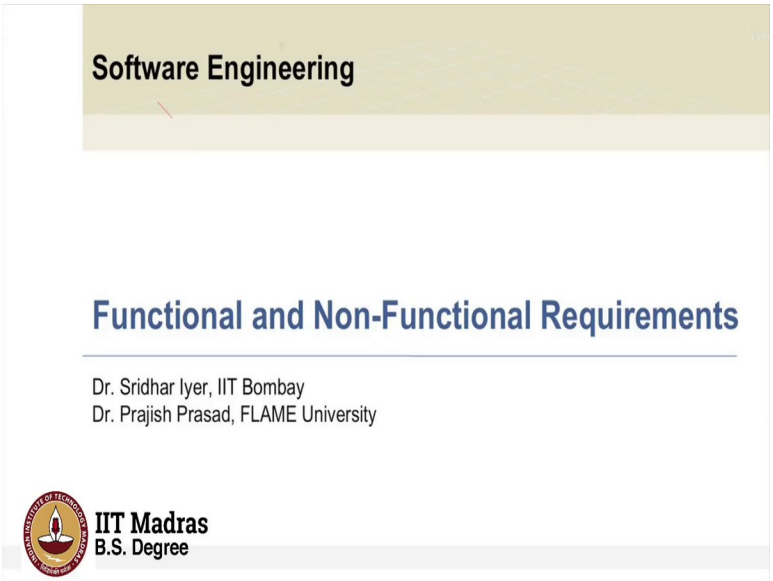


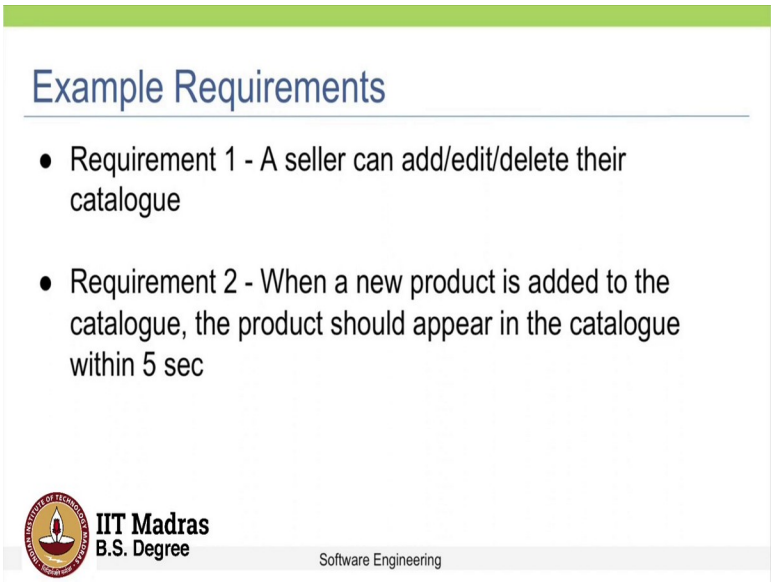
So in this lecture, we looked at behavior driven development or BDD, BDD ask questions about the behavior of an application before and during development, so that the stakeholders are less likely to miss communicate. We saw that the BDD version of requirements is user stories. We saw the roll feature benefit pattern or template.

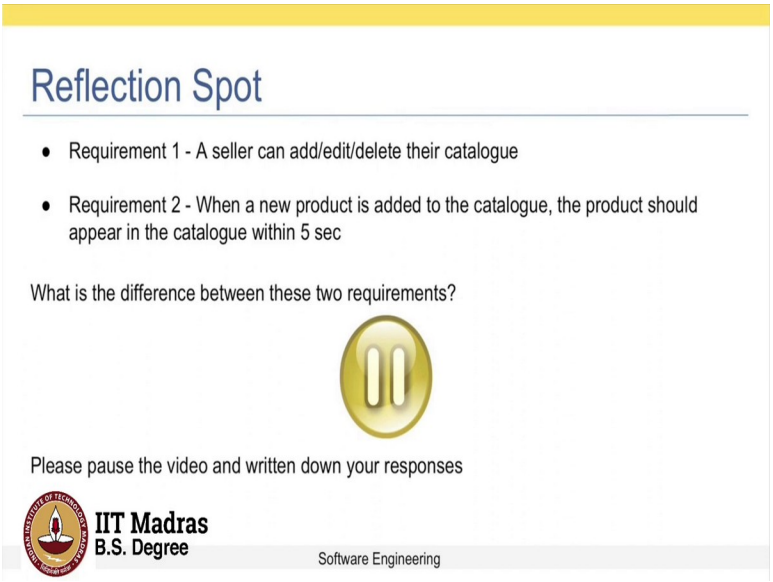
We looked at the characteristics of good user stories, that is smart user stories that it should be Specific, Measurable, Achievable, Relevant and Timeboxed. And we also looked at the benefits and drawbacks of user stories.

Software Engineering
Professor Sridhar Iyer
Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Professor Prajish Prasad
Department of Computer Science
FLAME University
Functional and Non-functional Requirements

(Refer Slide Time: 0:20)

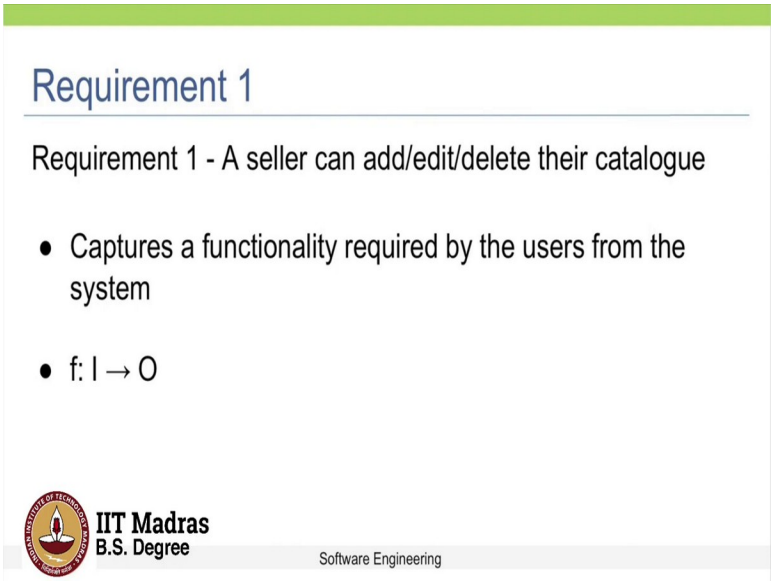


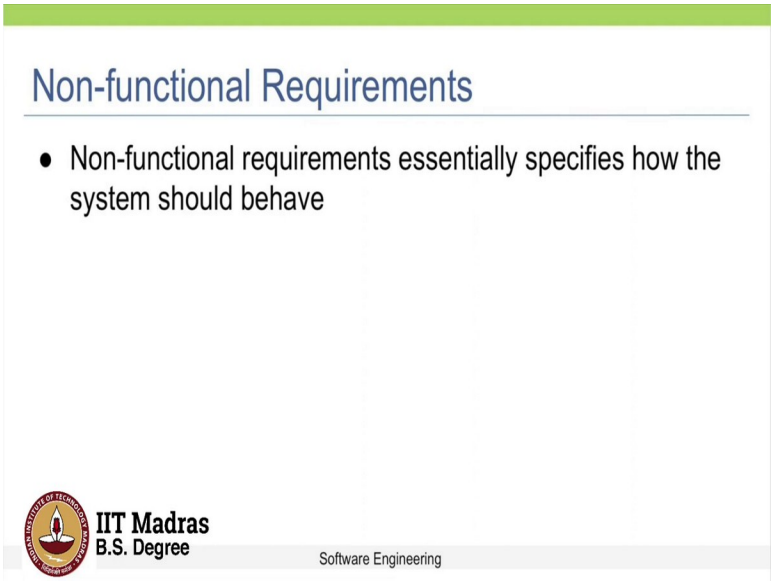


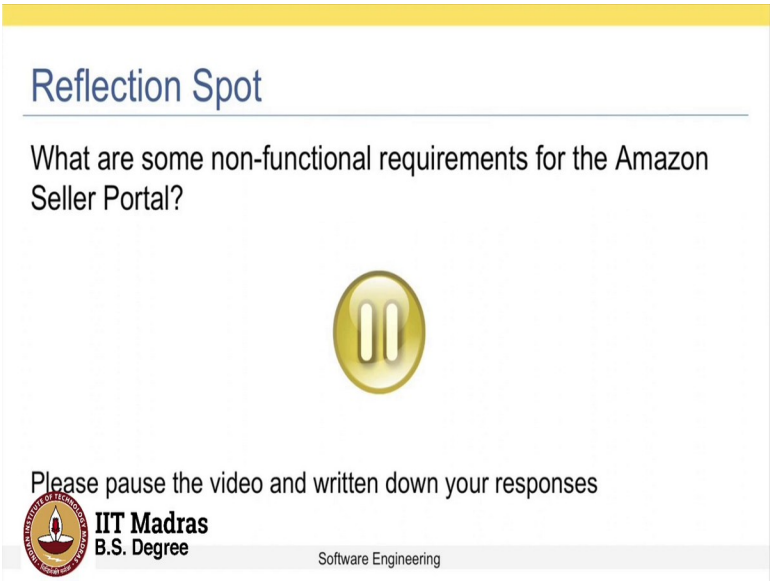


Professor Sridhar Iyer: In the previous video, we looked at how we can identify requirements using several techniques? Such as interviews, looking at the documentation, questionnaires, and so on. You might have noticed that not all requirements have the same characteristic, or similar type of characteristics. For example, consider these two requirements. A seller can add or delete items from their catalogue, this is one requirement. Second one is, when a new product is added to the catalogue, it must show up on the user's interface within 5 seconds. Can you say what is the difference between these two requirements? What is a nature of these two requirements? How was that different? Pause here for a moment, think about it, and then proceed.

(Refer Slide Time: 1:28)







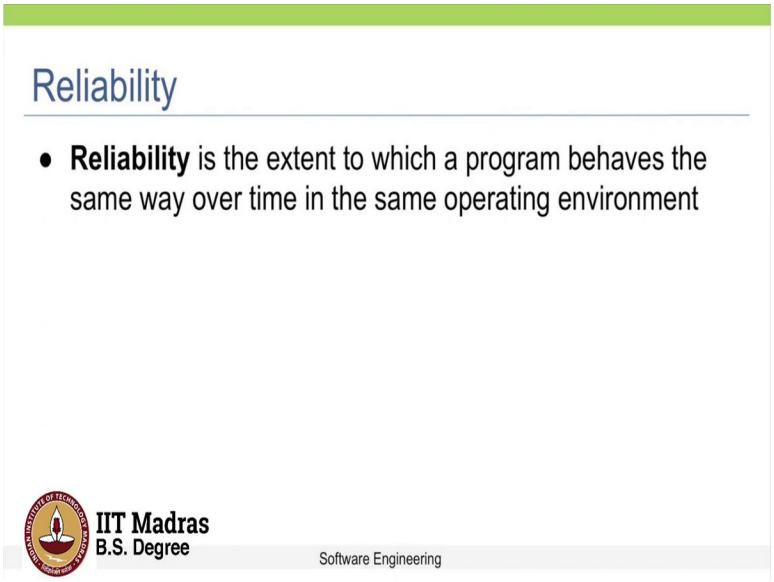
Professor Prajish Prasad: So, let us look at the first requirement. So, requirements 1 captures the functionality required by the users from the system. So, the user performs some actions, give some inputs, and based on that input, the system provides an appropriate output. So here, you can see that the user can maybe add, edit, or delete some things from their catalogue, and the output will be a result of these actions. So, these types of requirements are known as functional requirements. So, functional requirements can be considered similar to a mathematical function, where the function transforms an element in the input domain I to a value in the output O. So, the functional requirements of the system should clearly describe each functionality that the system would support along with the corresponding input and the output dataset.

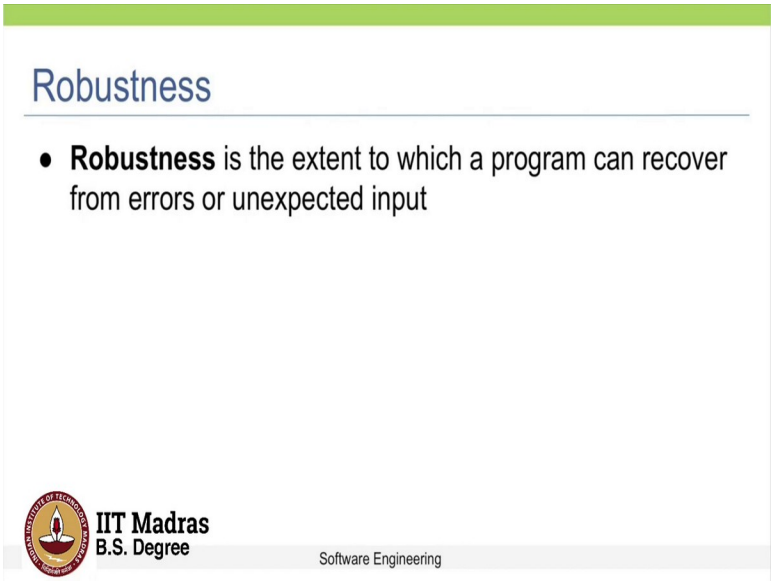
Professor Sridhar Iyer: So, coming back to the requirements, we have saw that requirement 1 of being able to add or delete a product from the catalogue is a functional requirement. The

requirement, second requirement that we talked about that is that the product should appear in the catalogue within 5 seconds. It is not a requirement that can be expressed as functions, that is as accepting some input and producing some output data. So, such requirements essentially specify how the system should behave, not what the system should do. So, such requirements are called non-functional requirements.

It is a little difficult to understand both these terms in the beginning, especially because we are used to the word non-functional interpreting the word non-functional in a different way. So again, let us take the example of the seller portal. We have talked about one non-functional requirement which was that the product update should appear within 5 seconds. What are some other non-functional requirements that you can think of for the system? Pause here, write down one or two non-functional requirements and proceed.

(Refer Slide Time: 4:11)



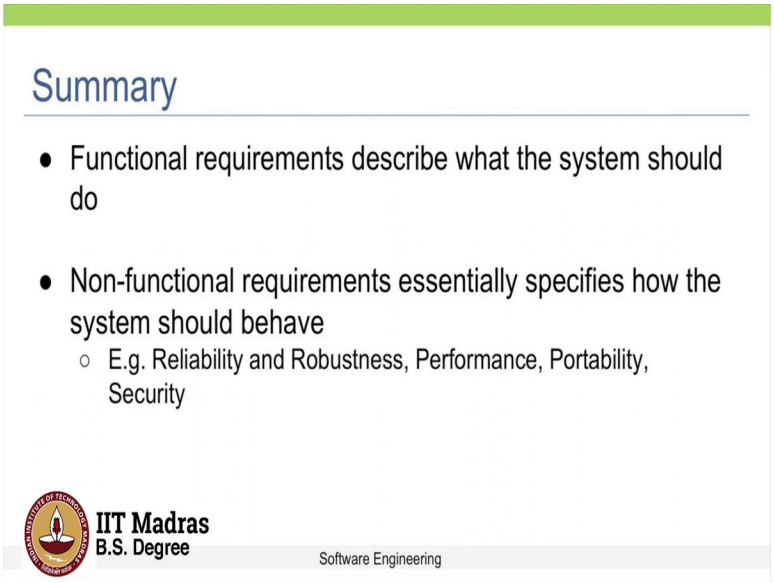


Professor Prajish Prasad: One type of non-functional requirement is reliability. So, a client would want the system to be reliable. So, reliability is the extent to which a program behaves the same way over time in the same operating environment. So, in the case of software, we need to ensure that the software does not crash often, otherwise users will get frustrated. For example, in the seller portal reliability is even more important. We must ensure that the portal handles all operations, especially when the inventory changes based on the buying and selling from customers.

Professor Sridhar Iyer: Now, in addition to reliability, we also want the system to be able to handle unexpected situations. What does it mean? For example, a user may just give some garbage as input, what is the system do now? So, such features, or non-functional requirements come under the category called robustness. So, robustness is the extent to which a program can recover from errors and unexpected input. For example, in the seller portal, we have to ensure that the service handles large numbers and high traffic.

When we specify requirements related to reliability and robustness, we are in a way ensuring that the system behaves the way we intend it to behave. And we are not exactly specifying what we want the system to do? Or we are not checking the functional requirements. So, that is the key between functional and non-functional requirements.

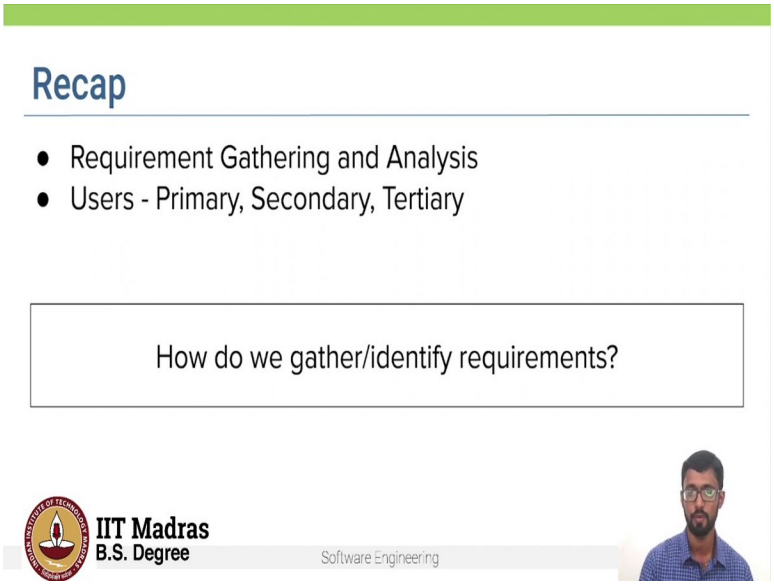
(Refer Slide Time: 6:10)



Professor Prajish Prasad: So, to summarize, in this video we looked at functional and non-functional requirements. We saw that functional requirements describe what the system should do? Whereas, we saw the non-functional requirements like reliability, robustness, they essentially specify how the system should behave? So, in addition to reliability and robustness, there are other non-functional requirements as well, such as performance, portability, security, interoperability, etc. So, we will not look into those in detail now, but it is important to consider the functional as well as non-functional requirements when we start building a software system.

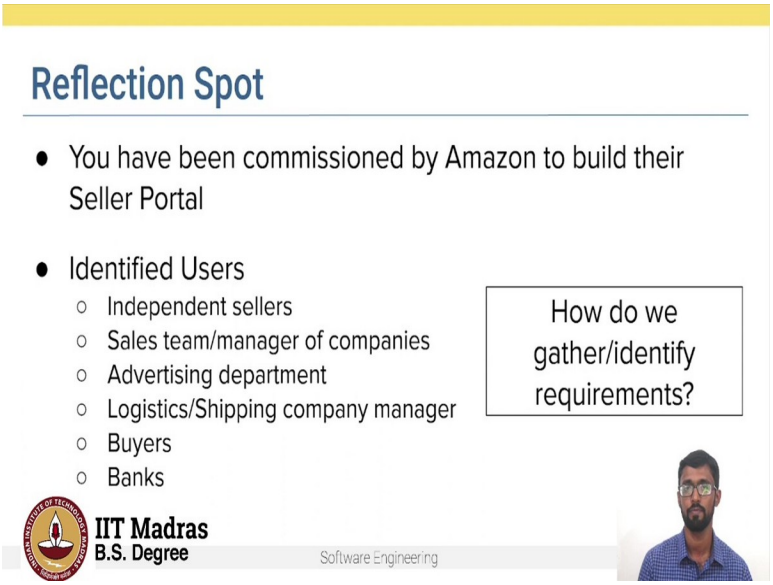
Software Engineering
Professor Doctor Sridhar Iyer
Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Doctor Prajish Prasad
Department of Computer Science
FLAME University
Identifying Users and Requirements

(Refer Slide Time: 00:28)



So, in the previous video, we looked at requirement gathering and analysis. And we also looked at how we need to consider primary, secondary and tertiary users of a system. Now that we have identified these different types of users, the question is now how do we gather or identify requirements from these users? Do we just talk to users? Or are there other ways to gather these requirements? So, let us reflect on this question for a moment.

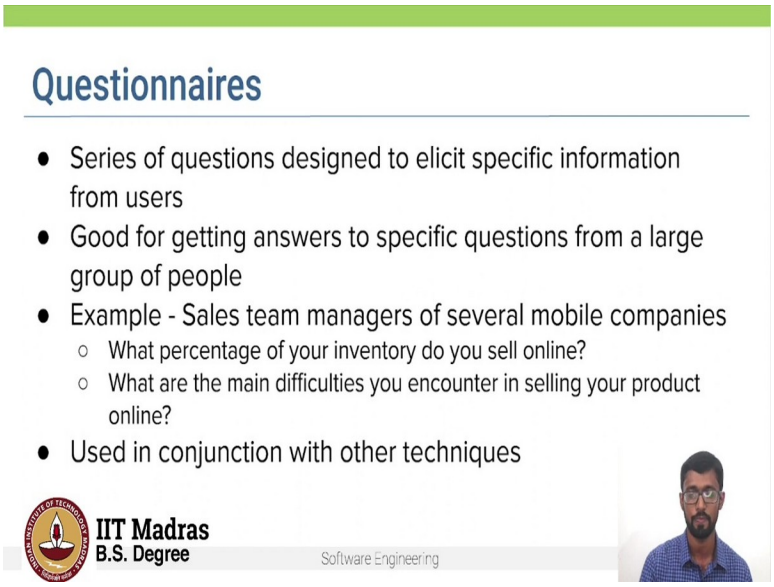
(Refer Slide Time: 1:04)



And let us take the example of the Amazon Seller portal that we looked at in the previous video. Let us say that it is the early days of Amazon and you have been commissioned by Amazon to build this Amazon seller portal. So, here are some of the users that we identified in the previous lectures. So, we have independent sellers, we have a sales team or a sales manager of different companies. We have people from the advertising department, we have logistics or shipping company managers, of course, we have buyers, and we have banks. So, these are different users of the system.

So, now the question is what are ways in which we can gather requirements from these different users for developing a seller portal? You can pause this video and think about these answers before proceeding.

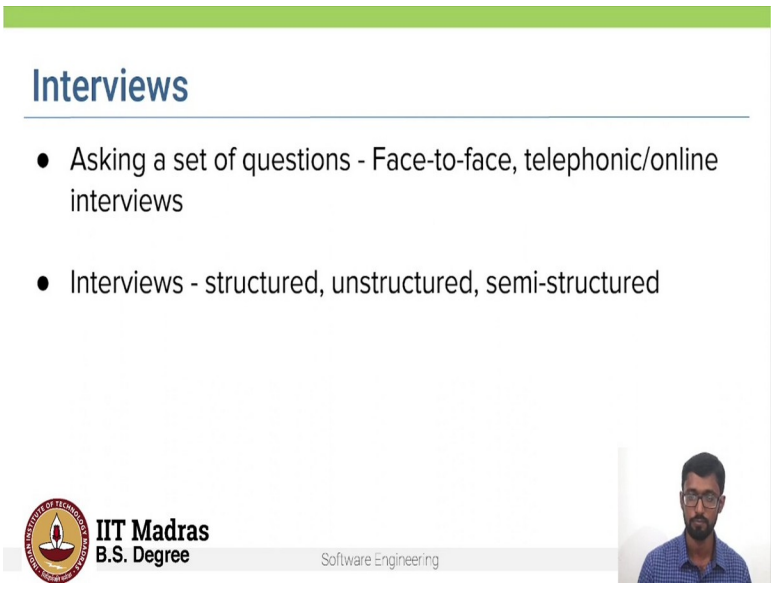
(Refer Slide Time: 02:18)



So, there are different requirements collection techniques. And let us look at them one by one. So, one type of requirement collection technique is what is known as a questionnaire. So, a questionnaire is nothing but a series of questions, which are designed to elicit specific information from users. And I am sure most of you would be familiar with questionnaires as you would have filled such questionnaires, as part of different courses and the program as well. And these questions can be of different types can be a simple yes or no question. Maybe you have to choose from a set of answers, or it can be a longer response or a comment. And questionnaires are good for getting answers to specific questions from a large group of people who are usually spread across a wide geographical area.

So, for example, you can send questionnaires to sales team managers of several mobile companies. And you can ask them questions like what percentage of your inventory do you sell online? Or what are the main difficulties you encounter in selling your product online? So, these answers to these specific questions can help us elicit requirements from different users. And an important thing to note is that this should be used in conjunction with other techniques, other requirement gathering techniques, which we will see the next slide.

(Refer Slide Time: 04:07)



So, another important requirement gathering technique is interviews. An interview, it involves asking someone a set of questions. It is often face to face but it need not be. We can use telephonic interviews or even online interviews to ask questions. And interviews are broadly classified as structured, unstructured or semi structured interviews, depending on how rigorously the interviewer sticks to a prepared set of questions. So, an interview can be

structured, which is nothing but a fixed set of questions. And the interviewer asks those questions and elicit responses.

And in an unstructured interview the interviewer does not really know what type of questions to ask, because they do not know what the user does or what exactly the needs are. So, in such cases, we can ask users broad questions and based on their responses, we can probe further into specific aspects.

(Refer Slide Time: 05:23)

So, what are the purposes of these interviews? One is, it helps us get or understand issues which people have. And it can be used early in the requirements gathering process to illustrate different scenarios. So, for example, we can conduct interviews with independent sellers, and ask them what are different platforms which you use to sell your products? This is a fairly open ended question. And they might give answers like, we use WhatsApp, Instagram and then we can probe them and ask what are disadvantages or advantages of these portals?

And they might say, well, I get customers it is easy to get customers, but then tracking orders, you know, tracking payments, tracking my inventory, what are different selling options, all of this is huge overhead. So, from, from these questions, we can elicit requirements, which involve tracking orders, tracking payments and all of these things.

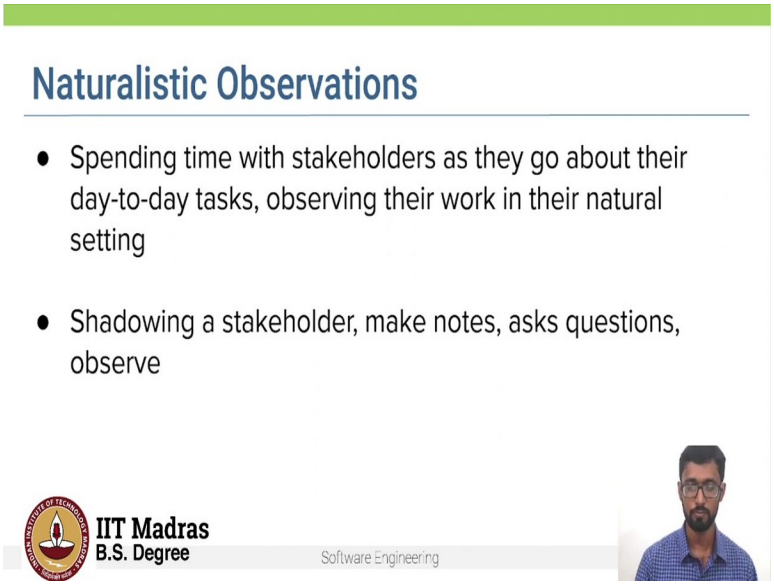
(Refer Slide Time: 06:39)

So, now, we looked at interviews and one drawback of interviews is that it offers only one perspective. And an alternative can be to get a group of stakeholders to discuss issues and requirements. And the advantages of using focus groups are one as it helps us gain consensus. So, if different groups of stakeholders if they highlight the same issue or the same concern, then we know that this is a key requirement, which we need to build into our system. And this is also useful for highlighting areas of conflict or disagreement. So, different groups of people might not agree. And this helps us get a nuanced understanding of the needs of different stakeholders.

(Refer Slide Time: 07:41)

So, let us take an example from the Amazon seller portal itself. So, we can conduct focus groups with different sales team managers from different verticals. And there might be different expectations from different industries. So, for example, a mobile sales team might be very interested in understanding what is the sales summary at the launch of a particular mobile whereas departments like furniture, they might be more interested in the sales summary of the last 30 days. So, from such focus groups, we can elicit requirements like such as the summary of the sales in the past x days or in a specific day. So, these are some requirements which can be elicited from focus groups.

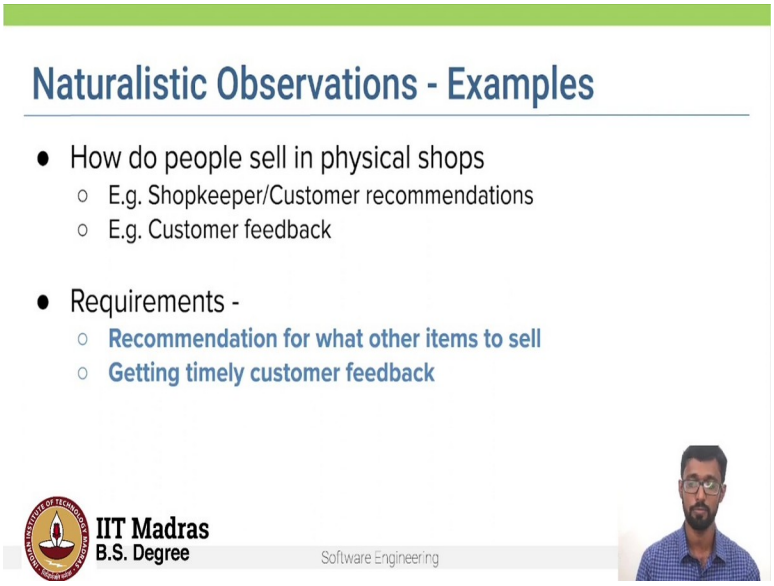
(Refer Slide Time: 08:36)



In the previous requirement gathering techniques, we looked at interviews, focus groups, in these techniques, we are asking people questions, but one thing to remember is that what people say is often different from what they do. And it is very difficult for humans to actually explain what they do or describe accurately how they do a particular task. For example, how do buyers buy online? What are they looking for in a product? This might not be very easy to articulate. And hence an important technique which we can use is observe users. It involves spending time with stakeholders as they go about doing their day to day tasks and observing their work in their natural setting.

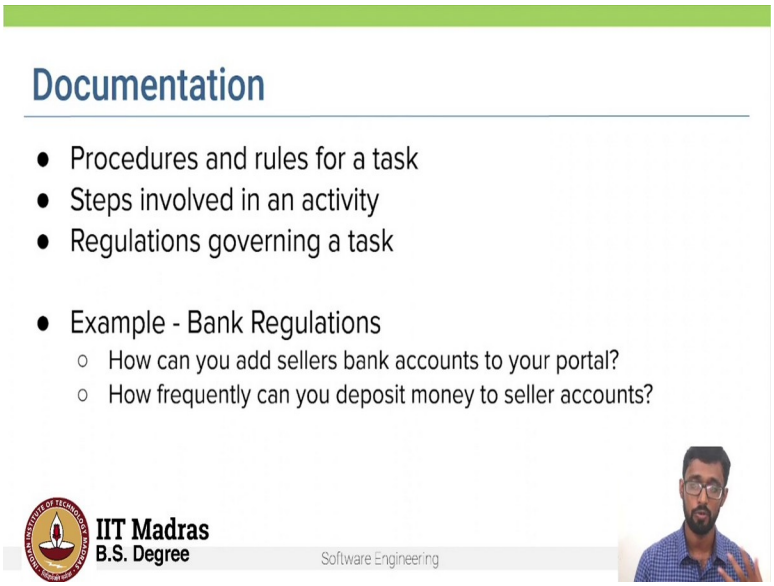
And this involves shadowing a stakeholder, making notes, occasionally asking them questions, but primarily observing what they do.

(Refer Slide Time: 09:44)



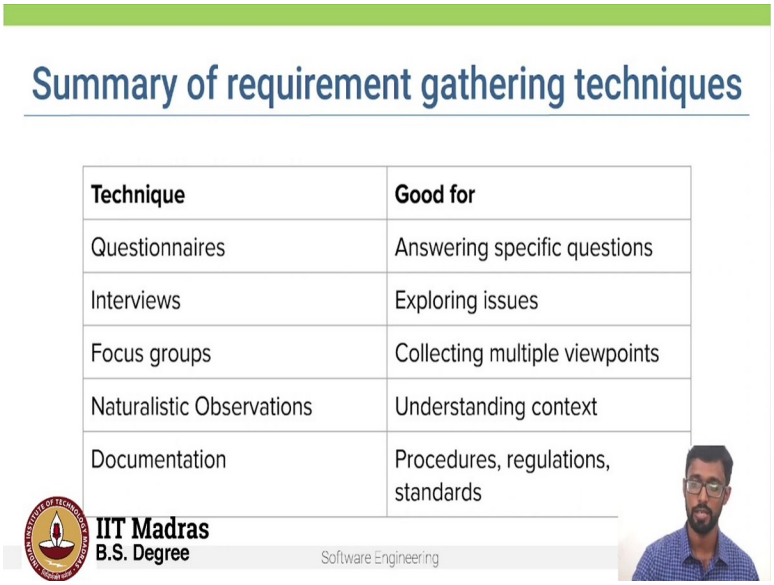
So, let us take the example of a seller, the seller portal. So, one way in which we can gather requirements is to look at how people sell in physical shops. So, for example, shopkeepers or customers they give recommendations, the customer might ask the shopkeeper for a new product or the shopkeeper might ask the customer to try a new product. Customer feedback is also very important, the shop owner, they appreciate feedback about a product which they sold to their customer. So, these can be translated into the online setting as well in the form of recommendations as well as getting timely customer feedback.

(Refer Slide Time: 10:43)



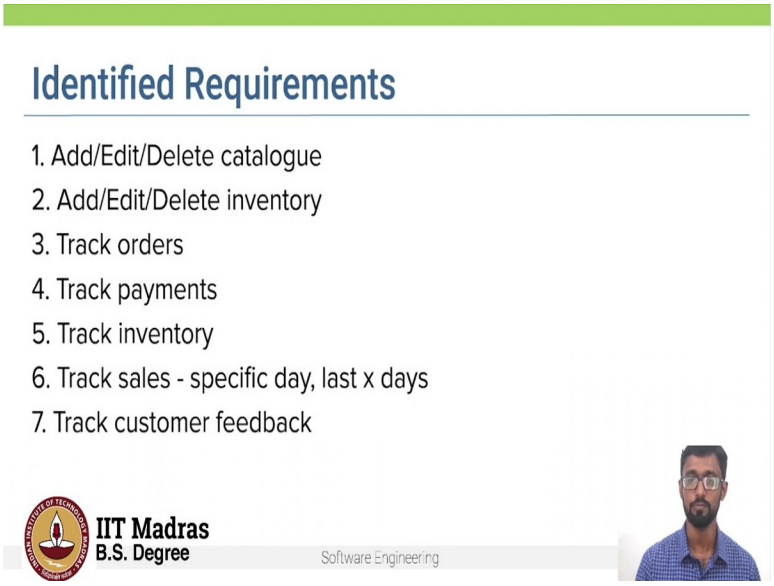
Another type of requirement gathering technique is documentation. So, there are places where which helps several procedures and rules for doing a particular task. And these might be written down in manuals or documents or there might be steps which are involved in performing an activity or there might be regulations which might be governing a task. So, in our case, in case of a seller portal, bank regulations are an important requirement to consider. So, for example, how can you add sellers bank accounts to your portal? Or how frequently can you deposit money to seller accounts? What are practices of other online marketplaces? All these can be gathered from different documents or different procedures and regulations.

(Refer Slide Time: 11:37)



So, here is a summary of the requirement gathering techniques which we saw till now. So, we looked at questionnaires, interviews, focus groups, observations and documentations. And each of them serves a specific purpose. For example, questionnaires are good for answering specific questions. And interviews are good for exploring issues and for which we do not really have the answers for. Focus groups are important for collecting multiple viewpoints. And observations are good for understanding the context in which a user performs a particular task. And then we looked at documentation which involves procedures, regulations and standards.

(Refer Slide Time: 12:30)



So, we looked at several requirement gathering techniques and using thought experiments on how we can gather requirements, we were able to identify several requirements for the Amazon seller portal. For example, we understood that a catalog and inventory are necessary and it is essential for us to track different things like orders, payments, the sales as well as customer feedback.

(Refer Slide Time: 13:07)

So, to conclude, let us look at some basic requirement gathering guidelines. So, first, we need to focus on identifying stakeholders needs. And we need to involve all the stakeholder groups, the primary secondary as well as the tertiary users, because each of these types of

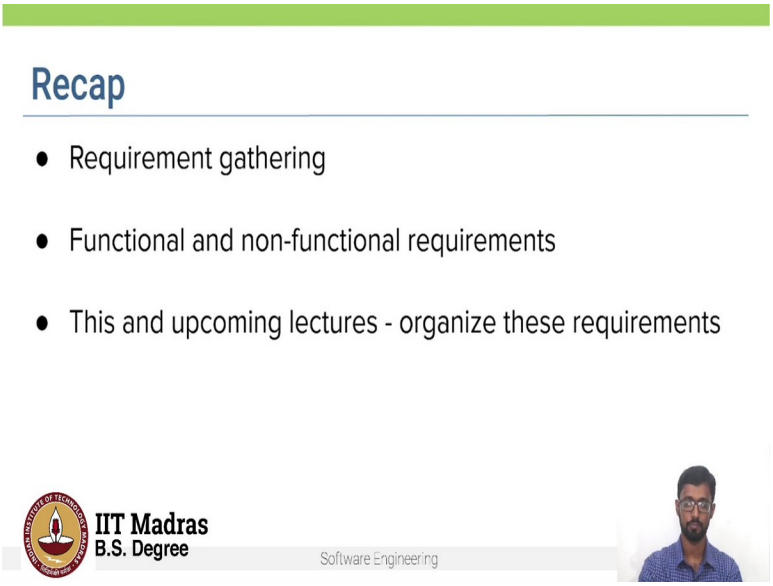
users have different needs and requirements. And it is advisable to use a combination of different requirement gathering techniques, mainly because different techniques have different purposes. And we need to use them according to our needs.

Another important point to consider is to run a pilot session, if possible to ensure that your data gathering session is likely to go as planned. Many a times especially for questionnaires, you might miss out on a particular question, you might have wrong options, you might be having missing options. So, it is difficult to go back and ask users to fill the survey again. And hence, it is important to test your gathering techniques before you actually go about and gather requirements.

And finally, data gathering is expensive and it is time consuming because you have to talk to different types of users. And once you get their responses you need to analyze them and all of this is a time consuming and effort consuming activity. And hence often at times we have to be pragmatic and we have to make certain compromises.

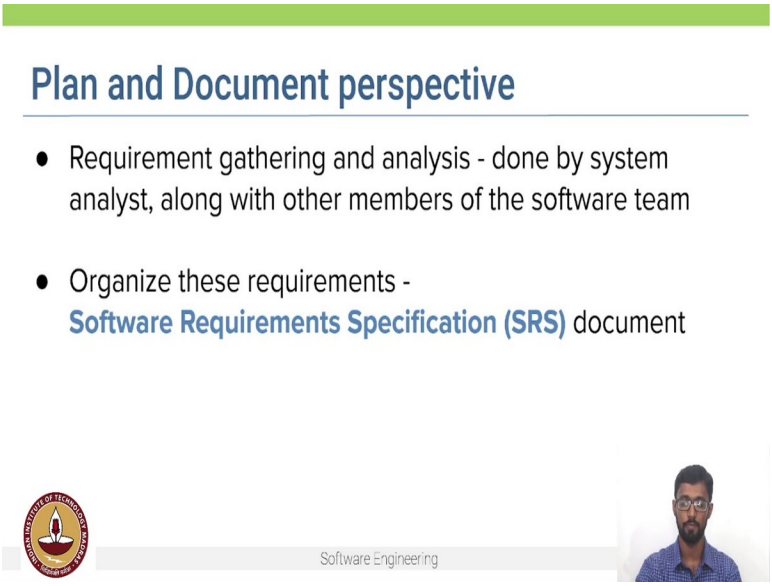
Software Engineering
Professor Sridhar Iyer
Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Professor Prajish Prasad
Department of Computer Science
FLAME University
Software Requirement Specification

(Refer Slide Time: 0:28)



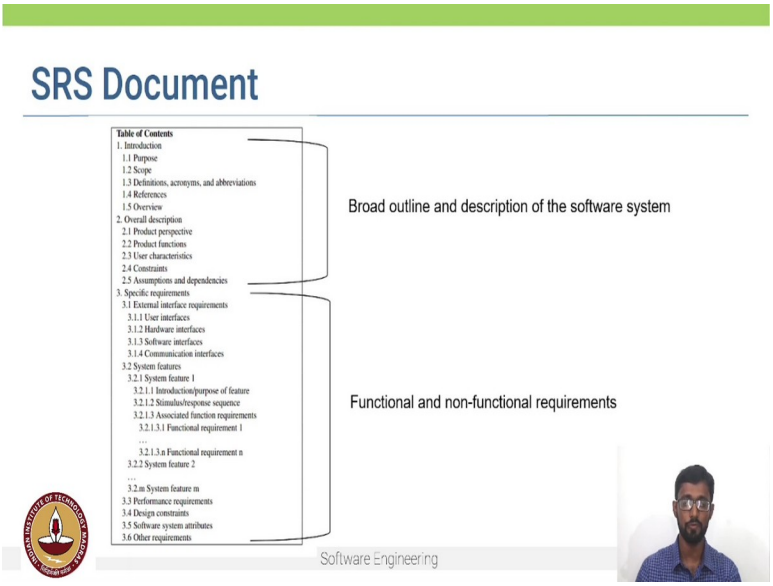
So, in the previous lecture, we looked at how we can gather and analyze requirements. And we categorize these requirements into two broad categories of functional and non-functional requirements. Now, another important point to consider is how we organize all these requirements that we have gathered and analyzed? So, in this and the upcoming lectures, we will look at means, by which we can effectively organize these requirements.

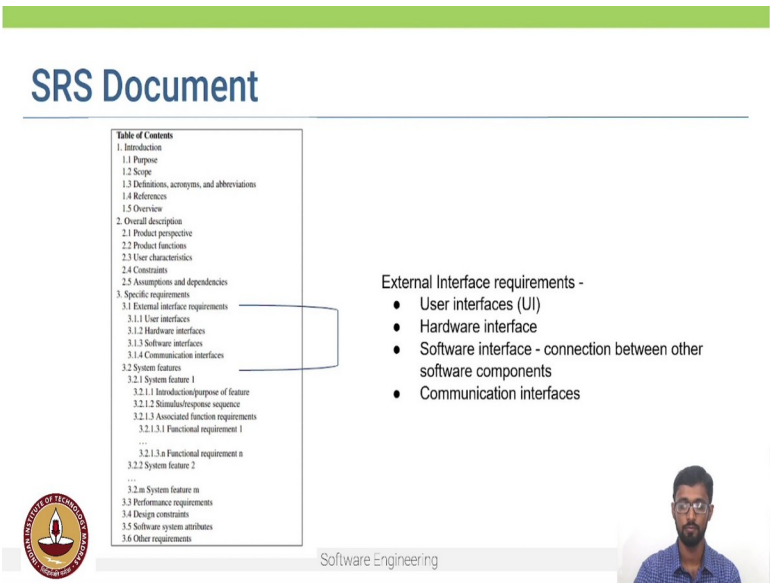
(Refer Slide Time: 1:05)

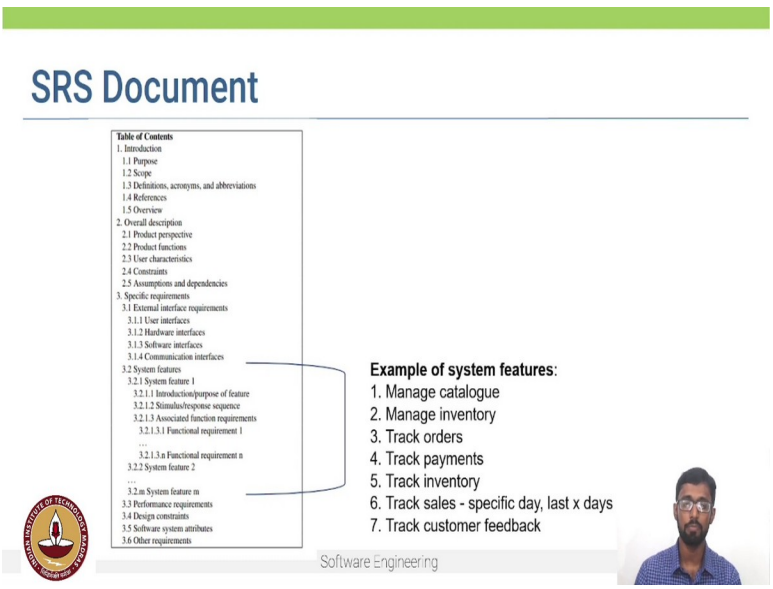


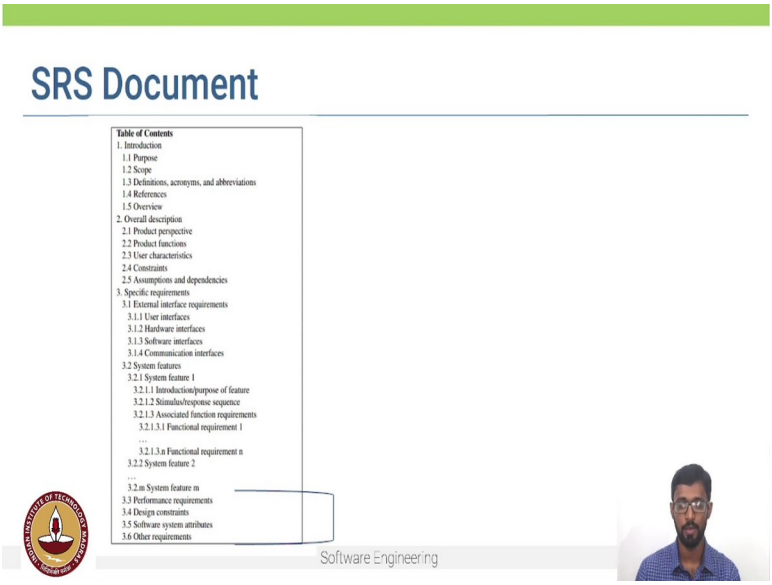
In the first week, we looked at two prominent software process models, the plan and the document model and the agile model and recall that in the planning document model, significant time is spent in planning and documenting the entire software process. And in the Plan and Document perspective, requirements gathering and analysis is usually done by a person known as a system analyst in consultation with other members of the software team. And after the analyst has gathered all the required information regarding the software to be developed, he or she starts to organize these requirements in what is known as the software requirements specification or an SRS document. And this document contains all the user requirements in a structured form. So, now, let us look at the standard format of an SRS document.

(Refer Slide Time: 2:17)









So, this is the standard structure of an SRS document. So, it has a table of contents which has several sections and subsections. Now, let us look at some of the important sections in the SRS document. Now, section 1 and 2, it describes the broad outline and description of the software system which we want to build. For example, it contains the purpose the scope, several definitions, acronyms and abbreviations which will be used in the software as well as the perspective, functions, constraints, assumptions, dependencies and so on. Now, section 3 of the SRS document contains the specific functional as well as non-functional requirements of the system.

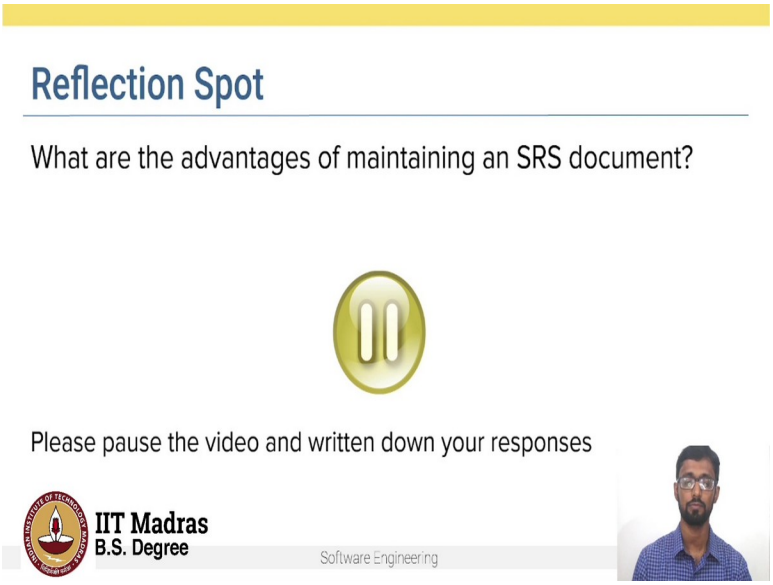
Now, let us look at Section 3.1. Section 3.1 refers to the external interface requirements. And here we specify things like what are the user interfaces? For example, what are the sample screen images, GUI standard, screen layout etc. Then we have hardware interfaces that is the interface between the hardware and the software such as the description of supported device types, the nature of data and control interactions between the software and the hardware. For example, in an ATM system, we need to specify the interface between the ATM hardware, the ATM card and the ATM software. Next is descriptions about the software interface which is the connection between this software and other software components.

For example, the database what is the operating system, what are other tools and libraries which will be used and so on? So, for example in the Amazon seller portal example, which we described in the previous lectures. So, the seller portal has to interface with the Amazon buying portal database. So, this is an important software interface which we have to specify. Then we have communication interfaces, which are any communication which is required by

the software example, sending an email or SMS notification about orders in the seller portal. So, that requires us to interface with an email server and such communication interfaces have to be specified. Now, when we look at Section 3.2, it outlines a broad high level function which is known as the system feature and the corresponding functional requirements for each of these system features.

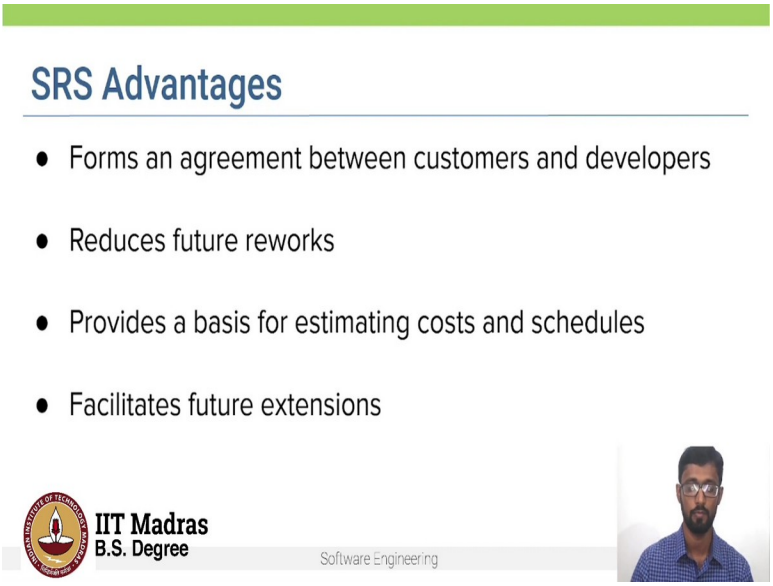
So, for example, taking the example of the Amazon seller portal, we identified several key requirements such as managing the catalogue, inventory, tracking orders, payments, inventory and all of those things. So, all of this forms specific system features which can be listed in the system features section. And for each of these system features, we add additional functional requirements and finally, in Section 3.3 to 3.6 contains details of non-functional requirements like performance, security etc. So, one thing to remember is that this is a guideline of how an SRS should be organized and it is generally not very rigid based on the organization and the context you are in modifications to the structure and the content are possible.

(Refer Slide Time: 6:59)



Now, that we have seen what an SRS document contains, let us reflect on why maintaining such an SRS document is important. So, what do you think are some advantages of maintaining such an SRS document? Please pause this video and write down your responses before proceeding.

(Refer Slide Time: 7:28)



So, let us look at some advantages of using an SRS. First, it helps to form an agreement between the customers and developers. So, after the SRS is prepared, the SRS document is also given to customers, the customer read it and after several iterations finally the SRS document is accepted. So, this helps the customers to form an expectation of what they will get from the developers and for the developers about what the customer expects from the software.

Second, it helps to reduce future reworks and SRS forces stakeholders to rigorously think about all the requirements before the start of the design and develop and this helps to reduce changes in later states. Third, an SRS provides a basis for estimating costs and schedules. So, in subsequent weeks, we will look at important tasks in managing a software project. So, estimating cost and schedule is an important software management task.

And, an SRS helps in estimating the size of the software which is nothing but a function of all the requirements of the system. And, based on this estimation, other estimation such as effort required and cost are estimated. And based on these estimates, the project manager can also create a schedule for the development of the software. And finally, an SRS can facilitate future extensions and it can serve as a basis for planning future enhancements of the software system.

(Refer Slide Time: 9:32)

So, in this lecture, we looked at what a software requirements specification document is, we looked at different parts and benefits of the SRS. However, there is a drawback of SRS which is a lot of documentation is required. And, this is okay if the requirements are fixed. In the next lecture we will look at the Agile perspective, which is the Behavior driven design, which can help in addressing this drawback of SRS in the planning document perspective.

Software Engineering
Professor Sridhar Iyer
Department of Computer Science and Engineering
Indian Institute of technology, Bombay
Professor Prajish Prasad
FLAME University
Week 2: Software Requirements
Requirement Gathering and Analysis

Professor Sridhar Iyer: In the previous week, we looked at the software development process. And we saw various models like the Waterfall Model and Agile Model. In this week, we will be looking at the first step in software engineering, basically, identifying requirements for a software solution. This is a very important step, because we need to spend sufficient time and effort in identifying such requirements and documenting, why? Because if we do not do this thoroughly, there will be a mismatch between what the user expects and what the software developer does. So, this week, we will focus on requirements.

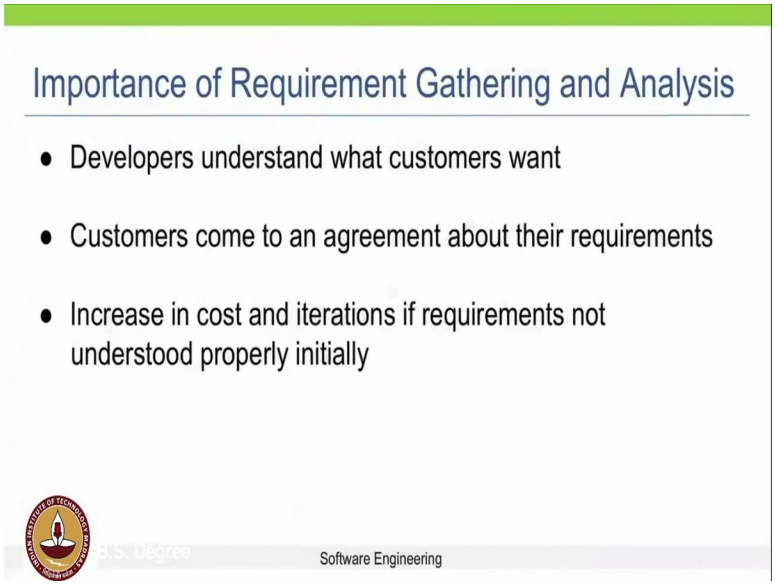
Professor Prajish Prasad: So, you might not be so convinced that identifying requirements are important. And you might also think why we need to spend so much time and effort in identifying as well as documenting these requirements? Cannot we just identify the requirements as we go about designing and developing the software system? So, let us reflect on this question.

(Refer Slide Time: 01:36)

So, learners, what are the disadvantages, if we do not spend time and effort in identifying and documenting requirements? You can pause this video and write down your responses before proceeding.

Professor Sridhar Iyer: Our vision of what the software should look like and behave is quite different from what the user has in mind. Even historically, many projects have had this problem, developers started to implement something while the users actually were expecting something else. So, there was a mismatch of expectations. What is the cost involved here, you spent a lot of time, effort and money in developing something, which was not what your customer asked for, so. So you have to make changes. So, there are more iterations in the software development process. And this pushes up the development costs very significantly.

(Refer Slide Time: 02:47)

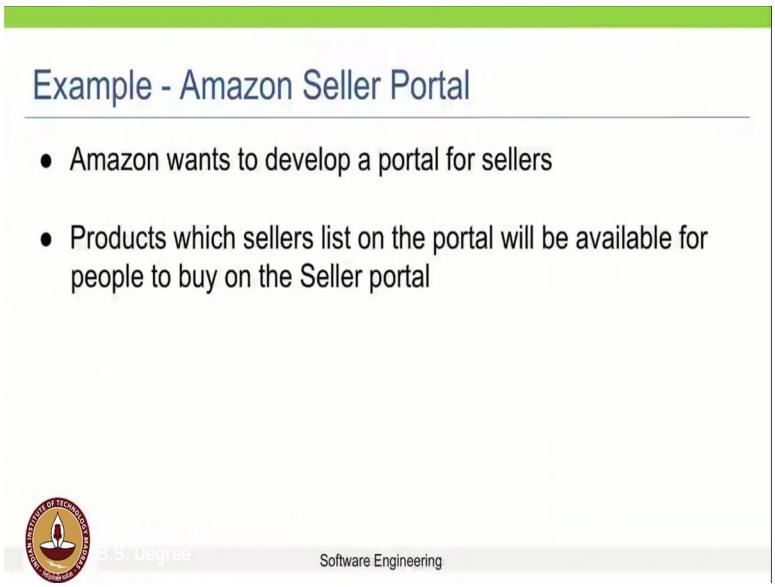


So, to summarize, the importance of requirement identification and analysis are two fold. One is that we understand developers understand what the customers want, of how the software should look and behave. And two, the customers agree that the developers are understanding is along the lines of what they are expecting. Otherwise, the cost, the time, the effort involved in developing the software can increase significantly. So, now that we have established that requirements are important, how do we go about it? Here, the first thing I need to know is what does my customer want from the system?

Again, remember, the customer is not always the end user. The customer can be an internal user, customer could be somebody from a different team. So, I need to ask the customers, what do they want? And then I look at what they want, analyze them, in order to plan how do I develop my piece of the software. All of this may sound fairly abstract at this point. And we will look at an example later. But the point to keep in mind here is that first, teams gather requirements. Then they analyze them. And only after that, they think about how to develop the software.

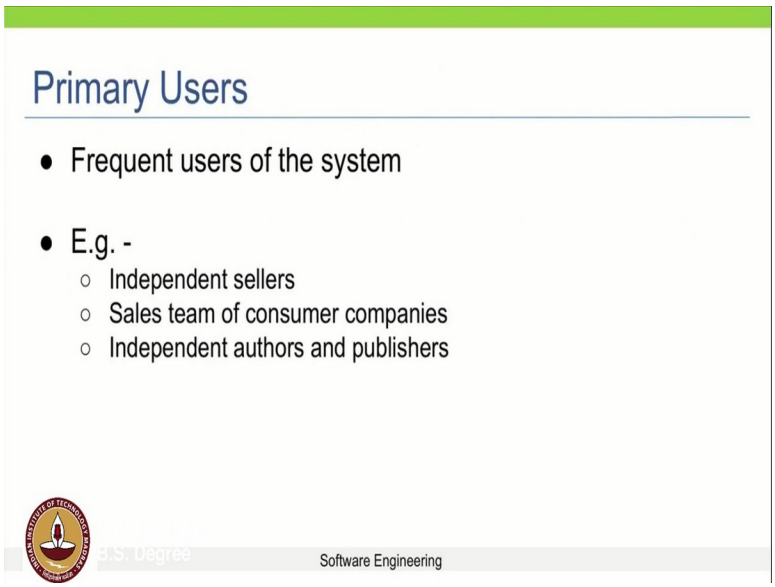
Let us start with requirement gathering. That seems very straightforward. Just go ask clients what they want, and we get a bunch of requirements. In reality, it is not that simple. First of all, there are different kinds of people who are going to be your customers in the system. All these users are called stakeholders of the system. So, when you introduce a software system, it influences different types of users. They may belong to your own organization, other organizations or end users and they may have different roles and profiles. So, what is required is to be able to understand the requirements of all these users in a broad, holistic manner before diving into designing the software. Let us look at this with an example.

(Refer Slide Time: 05:20)



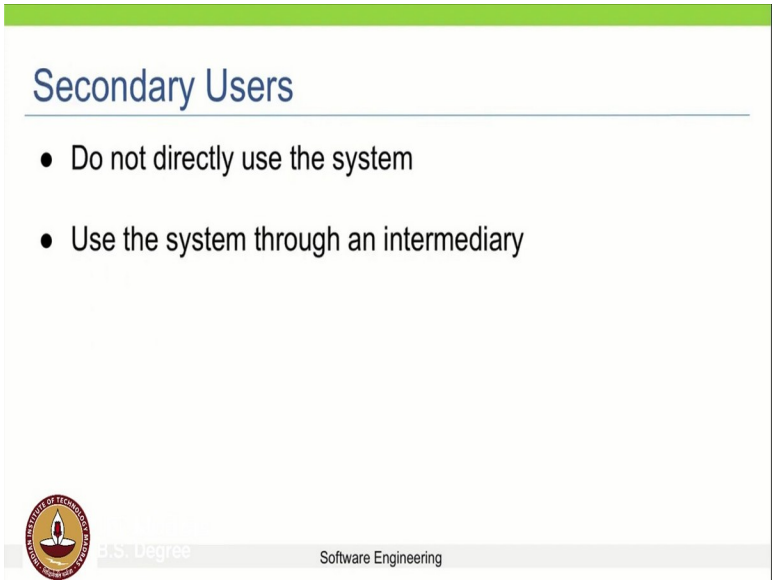
Professor Prajish Prasad: So, let us take an example from Amazon itself. And let us say Amazon wants to develop a portal for people who want to sell their products online. The products which sellers list on the portal will be available for people to buy on Amazon.

(Refer Slide Time: 05:41)



So, let us think about the frequent users of this seller portal. And these users are known as a primary users of our system. So, who do you think are primary users of this Amazon seller portal? Well, they can be the independent sellers who sell specific products. Primary users can also be the sales team of consumer companies, like those who sell electronics like mobiles, computers, etc. Even independent authors and publishers who want to sell their books online can be the primary users of this seller portal system.

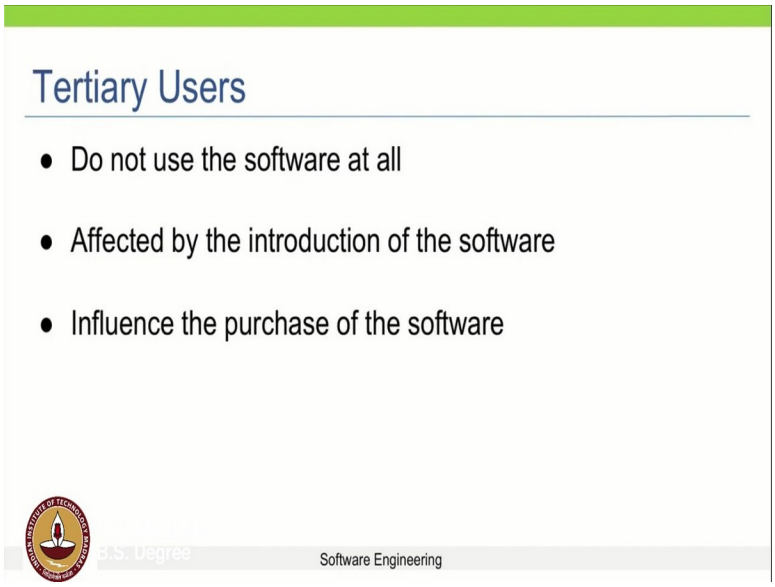
(Refer Slide Time: 06:29)



Then there are other types of users who are known as the secondary users. And these users, they do not directly use the system, but use the system through an intermediary. So, can you think of some secondary users of this seller portal? So, earlier, we mentioned that the sales

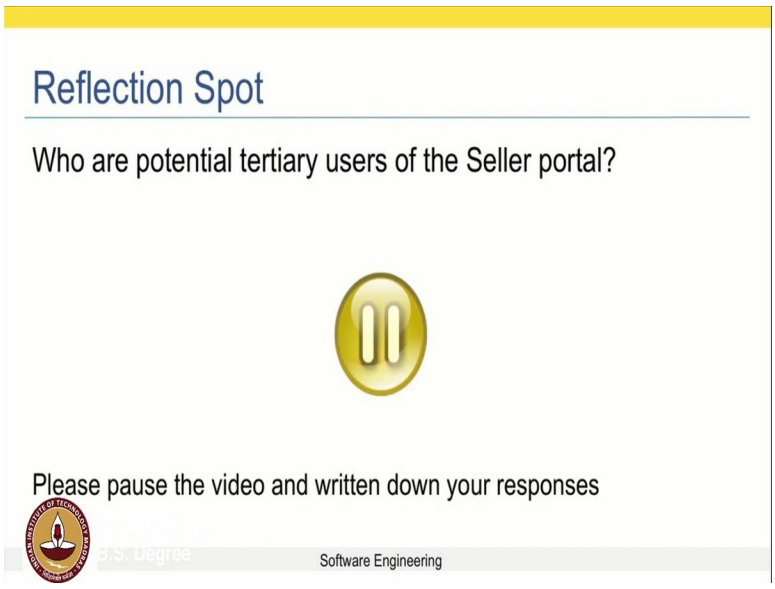
team, they can be primary users. And such sales teams usually have managers who periodically want to check the sales numbers, the profits and margins, etc. So, the sales managers can be the secondary users of this Amazon seller portal.

(Refer Slide Time: 07:16)



And then there are users who do not use the software, but are affected by the introduction of the software or they even influence the purchase of the software. So, these users are known as the tertiary users of the system.

(Refer Slide Time: 07:35)



So, learners, can you think of tertiary users of this Amazon seller portal? Who do you think our users who do not use the system at all but are affected by it in some way? You can pause

this video and write down your responses before proceeding. So, who are the tertiary users? Well, logistics shipping companies, they are responsible for delivering items to buyers. So, they are potential tertiary users of the system. Another type of tertiary users can be banks. So, the portal has to interface with banks to ensure that timely transfer of money happens to buyers based on the sale of their products.

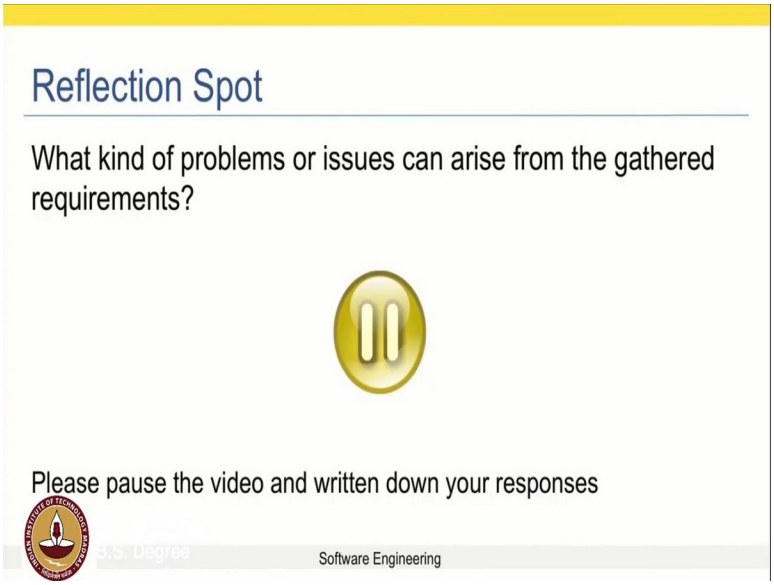
Whatever people who buy on Amazon, even they are tertiary users. They are buying depends on what products sellers sell on the portal. So, all of these users are possible tertiary users of the system.

Professor Sridhar Iyer: We saw that there are different users of the system. And we have to seek requirements from these users. So, how do we go about it? Do we randomly tap of users and ask them? Or do we conduct systematic interviews? There are many ways of doing this. In many cases, a basic interview might be sufficient. But there are also ways like studying existing documentation, having focus groups of different categories of users and also by observing what users are doing with the system or what do they need. We learn about all this in the next video.

Professor Prajish Prasad: So, now let us say we have gathered these requirements and now the next step is to analyze them. So, how do we go about doing that? So, during requirements analysis, one analyses the gathered requirements to form a clear understanding of the exact customer requirements and to way it out any problems in the gathered requirements. And these problems can arise because we will be gathering requirements from various stakeholders and from different sources. So, let us take the example of the Amazon seller portal which we discussed earlier.

Let us say we got a bunch of requirements from different clients, different stakeholders. So, now what kind of problems or issues can arise from the requirements gathered? Let us reflect on this question for a moment.

(Refer Slide Time: 10:47)



So, what kind of problems or issues can arise from the requirements gathered? Pause this video and write down your responses before proceeding.

Professor Sridhar Iyer: One issue can be that requirements are not clear. They might be ambiguous. What does this mean? For example, the requirement of independent sellers may be that they need to manage their inventory. But this is ambiguous because what does manage really mean? It depends on how you interpret it. I may interpret manage as being able to add one more product, you may interpret manage as being able to modify the price of a product or some other feature of the product. Somebody else may interpret manage as in a third way, say delete some products, add some new products and so on.

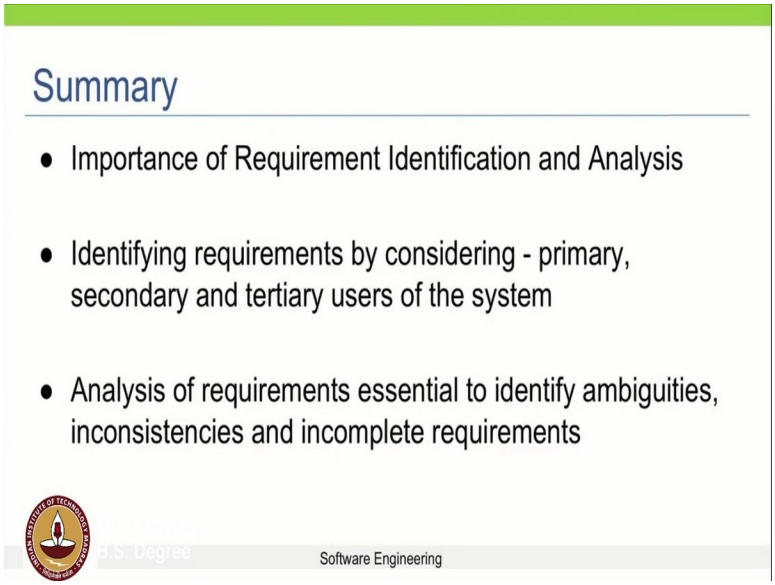
So, the point is that there are several interpretations possible. And developers will build a feature based on their own requirement. So, what we have to ensure is that the idea of what is required is the same both from the user's perspective and the developer's perspective. Another issue can be that the requirements are inconsistent or sometimes contradicting each other.

For example, if you think of the Amazon portal, it may be that independent sellers may want their payment credited every week. On the other hand, a banking system may say that the payment can be credited only bi-weekly. So, these two requirements may actually conflict with each other while building the system. So, as developers, we need to resolve these requirements in one way or the other. For example, you might say that for independent sellers, we might tell them that this is all that can be provided as a feature.

Once people know what the constraints are then you can go ahead with building the software. The third issue could be that requirements are incomplete. What does that mean? It means that some aspects of the implementation may have been overlooked, simply because neither the customer, nor the developer were able to imagine the all the details of the system before beginning to develop it. For example, independent sellers may mentioned that they want to be able to track orders. What does track orders mean? Does it mean that you want to track which orders are delivered? Or does it mean that you want to track the order while it is in transit? Or does it mean that you want to keep track of what orders have been returned, etc?

So, which type of orders does the seller want to track is an important information that the developer needs before being able to proceed.

(Refer Slide Time: 14:14)



To summarize, in this video, we have seen the importance of Requirement Identification and Analysis, identifying requirement by considering primary, secondary and tertiary users of the system. Analysis of requirement is essentially to identify anomalies, inconsistencies and incompleteness among the requirements. Once we have done this, we are in a better position to start designing the system.