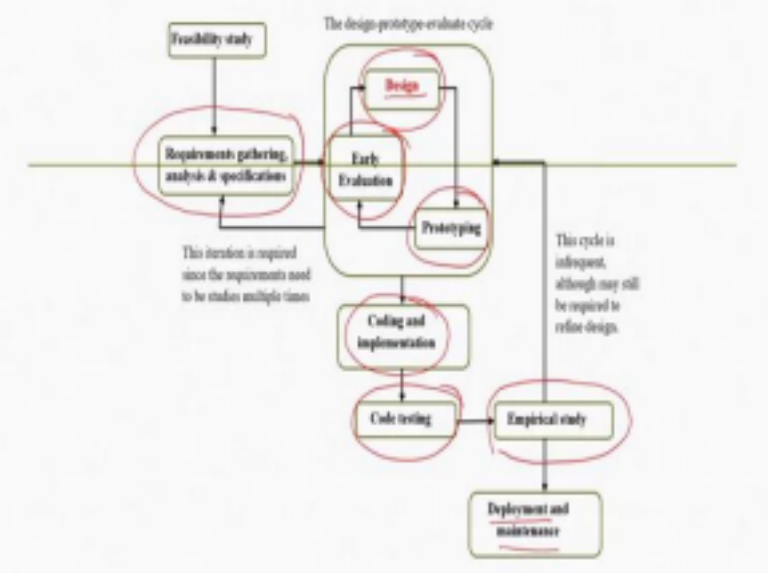
Design & Implementation of Human-Computer Interfaces
Dr.Samit Bhattacharya
Department of Computer Science & Engineering
Indian Institute of Technology Guwahati

Module No # 05
Lecture No # 23
Case Study on DFD & ER

Hello and welcome to the NPTEL MOOC's course on design and implementation of human computer interfaces lecture number 20 where we are going to discuss about case study on the concepts of DFD or data flow diagram and ERD or entity relationship diagram. Before we start, we will have a quick look at the interactive system development life cycle and where we stand. So, as you may recollect in an interactive system development life cycle there are several stages. **(Refer Slide Time: 01:20)**



We have requirement analysis gathering and specification stage. Requirement Gathering analysis and specification stage this is followed by a set of stages which form a cycle which is design, prototyping and early evaluation these 3 stages together constitute a cycle design prototype evaluate cycle. Now here if you recollect, we talk of design at 2 levels one is the design of the interface and interaction.

For that particular design, we need to prototype and get it evaluated to understand usability issues. If after evaluation, we find some issues we may need to go for refinement of the design then again, we prototype again evaluate, and in this way, it forms a cycle. Once, we arrive at a

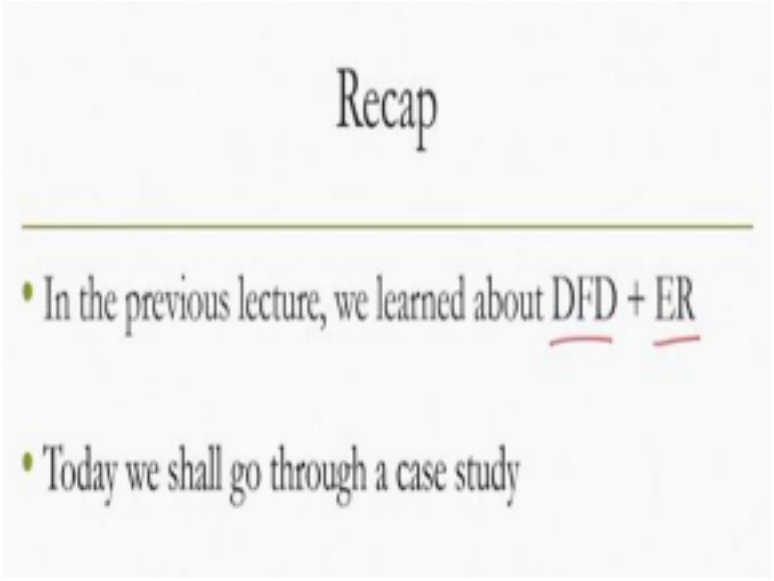
stable interface design then we go for implementation of the design. The first step of implementation is system design.

So, the design also implies system design. So here, with the name design, we are actually referring to both interface design as well as system design. For system design, of course, we do not require prototype and evaluation stages. So, once we arrive at a stable interface design, we go for the design of the system where we concentrate on modular design and there are several ways to express the design.

As we mentioned earlier, broadly 2 such ways are there one is a function-oriented approach and one object-oriented approach. In the function-oriented approach, we generally make use of DFD or data flow diagrams and ERD or entity relationship diagrams to express our design. Once the design is available, we go for implementation through coding which is the next stage. Then, we need to test the code. Program testing which follows the coding stage. After code testing, we get an executable system free from bugs.

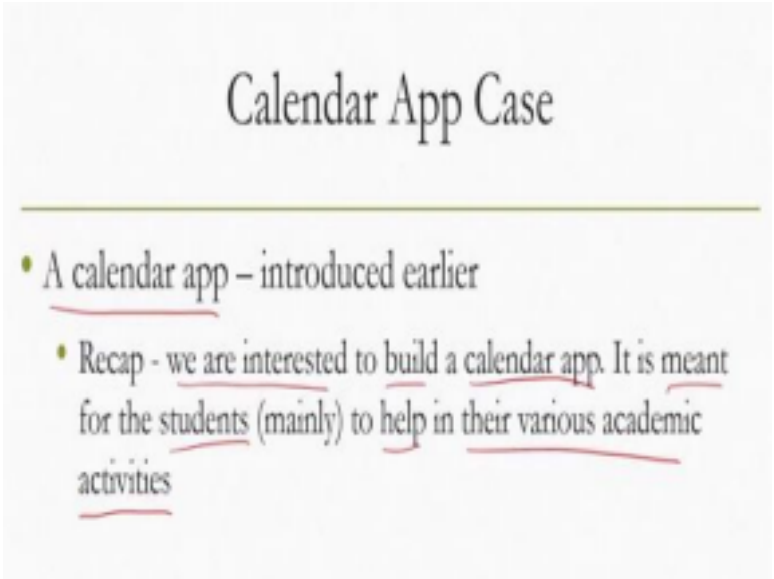
However, we still do not know about the usability of the system which we, evaluate in the next stage that is an empirical study. After the empirical study, we get to know of usability of the system. So, after this stage, we can expect a system that is executable as well as usable. This is followed by deployment and maintenance. So, these are the stages of an interactive system development life cycle and some of the stages from the cycle. That means they are repeated frequently and some should not be repeated frequently.

(Refer Slide Time: 04:43)



In the previous lecture, we learned about in details the idea of DFD or data flow diagram and ER or entity relationship diagram. So, whatever we learned in the earlier lecture, we are going to learn it further in terms of one case study which we will discuss in this lecture. So, we will go through a case study in this lecture. First, let us introduce the case that we are continuing from the earlier part of the course.

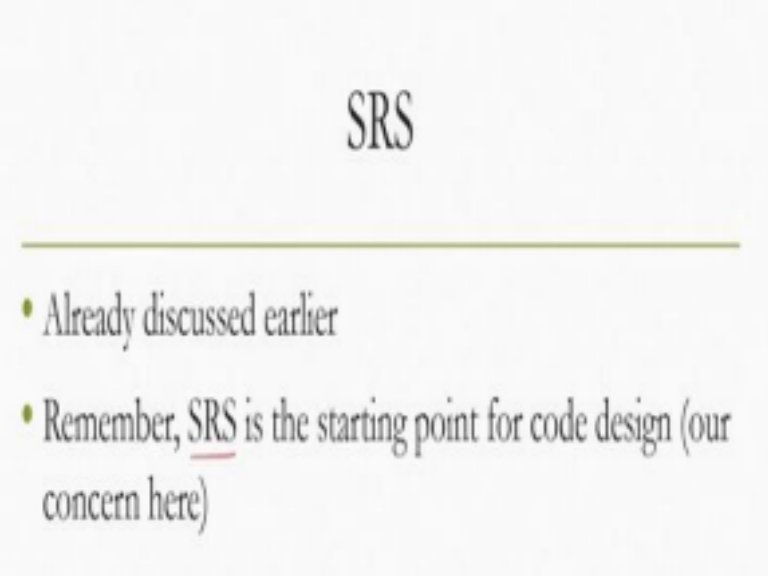
(Refer Slide Time: 05:22)



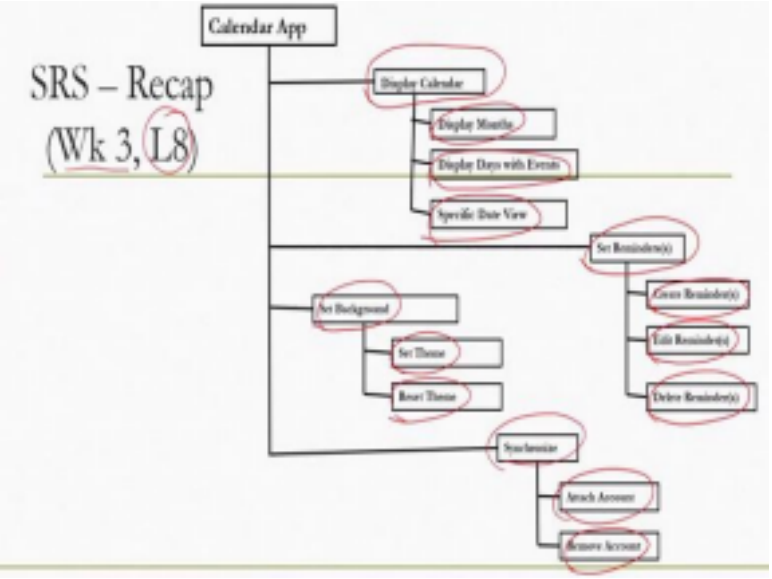
That is a calendar app or calendar application. We have already introduced the app earlier and discussed other development life cycle stages with respect to this app. Just to recollect what this app is all about. So, we are interested to build a calendar app which is primarily meant, to be used by students to help in their various academic activities. So, that is the idea of the app. So,

earlier we have seen how to create a requirement specification document that is a software requirement specification document or SRS for this app.

(Refer Slide Time: 06:13)



So, as we discussed in the previous lecture, the starting point of our system design activities is the SRS document. From the SRS document, we start designing our system. So, let us have a quick look at the SRS that we discussed for the Calendar app in an earlier lecture. **(Refer Slide Time: 06:38)**

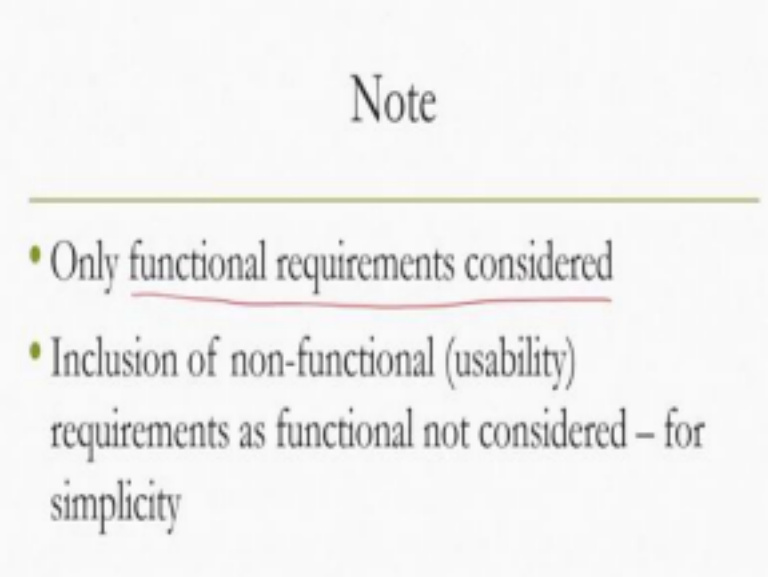


Particularly, lecture number 8, which we covered in week 3. So we will just briefly go through the various requirements that we have identified for this app. If you recollect, whenever we are trying to represent an SRS, the first thing is it should be hierarchical for improved manageability

of the code. So, we have created such a hierarchy. So, in the hierarchy at the top level, there are several requirements.

So, we have created such a hierarchy. So, in the hierarchy at the top level, there are several requirements. One is a display calendar under which come 3 sub-requirements or sub-functions, display months, display days with events, and a specific debt view. Then, we have another top level function which is to set a reminder, under which there are sub-functions create a reminder, edit reminder and delete reminders.

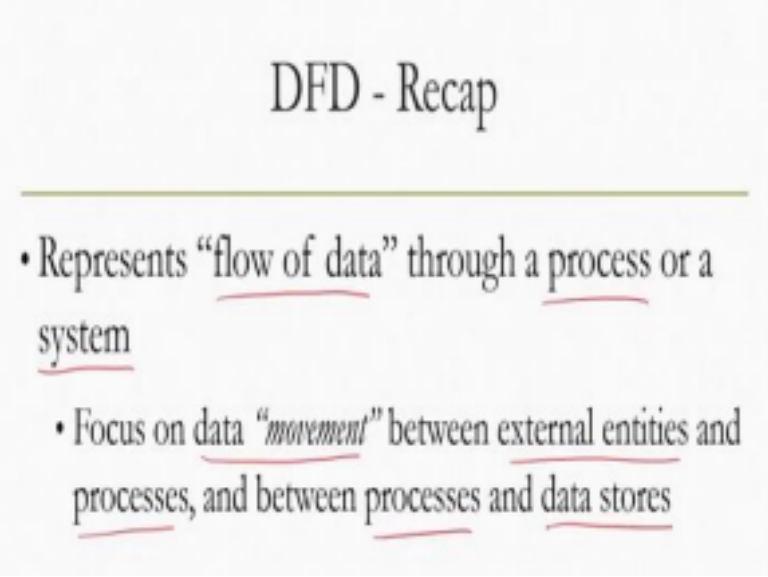
A third top-level function is set to background, under which comes set theme and reset theme. Finally, there is a fourth level. The fourth top-level function is synchronized, under which comes to attach an account and remove the account. So, here we will be showing only the hierarchy for a quick recap. You must remember that this hierarchy is accompanied by a detailed description of each of these functions in the SRS. In terms of their input, output description, and so on. **(Refer Slide Time: 08:25)**



Also note, that in this particular SRS that we have seen only functional requirements are considered. If you may collect earlier, we also saw how to convert some non-functional requirements into functional requirements and add them to the hierarchy. However here, for Simplicity we will be ignoring that part and we will be focusing only on the functional requirements. So, given this SRS our intention is to come up with a design and express the system design in terms of a DFD or data flow diagram which is a graphical language to express

design ideas. Before, we start discussing the DFD let us quickly recap what we have learned about DFD.

(Refer Slide Time: 09:19)

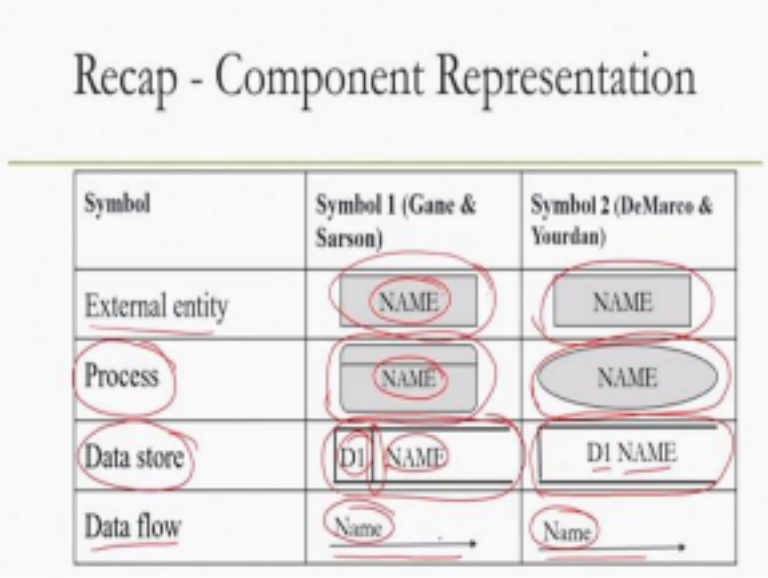


So, DFD represents flow of data, through a process or a system. So, here the focus is on data movement between external entities and processes and between processes and data stores. So, our primary concern is how the data flows between different stakeholders of a system including external entity, processes. So here, our main intention is to represent data flow between different components of a system and stakeholders that include external entities processes and data stores. **(Refer Slide Time: 10:07)**

So, what are the components of a DFT or data flow diagram? So, we talked about a few

components, primary components. The external entity which; acts as a source or sinks of data, processes, data stores, and finally the data flow. These are the primary components of any data flow diagram.

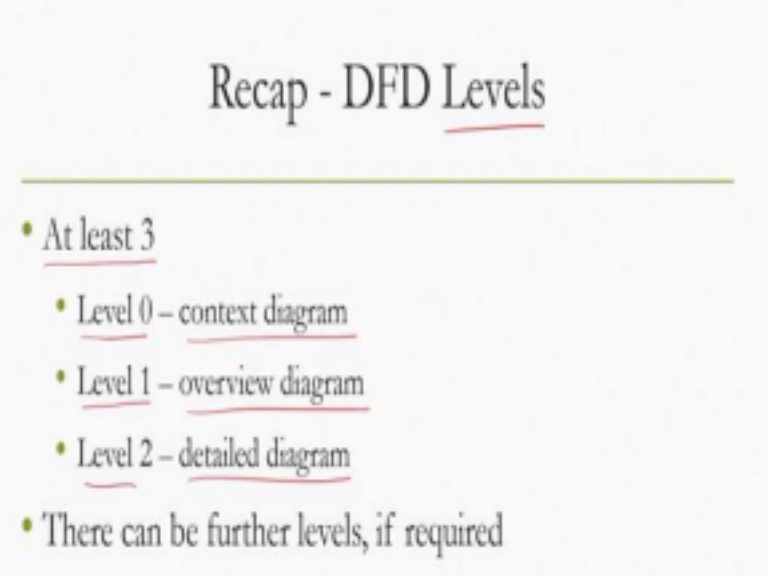
(Refer Slide Time: 10:37)



So, how do we represent them? Remember that we mentioned about 2 conventions. So, you can follow either of those 2. So, for external entity one convention tells you to represent the external entity with a rectangle, having the name of the entity inscribed in it. In the other convention also, the same symbol is used. For process, for representation of process, there is some difference.

In one case, we get one particular symbol with a name inscribed inside it. In another case, we get another notation. Data store also, there is a minor difference between an open-ended rectangle in one case with identifier and name separated by a vertical bar. Whereas in; another case we have the open-ended rectangle and identifier name but no vertical bar to separate them.

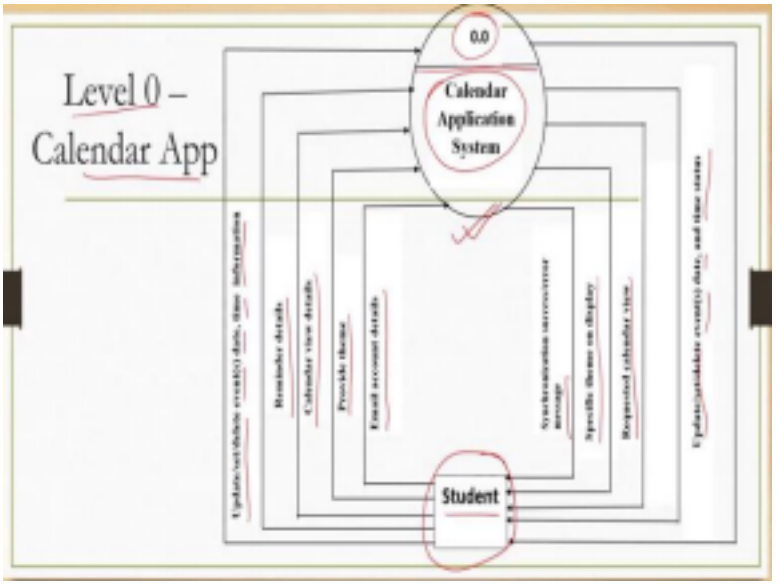
For data flow, the notations used in both the cases are the same and arrow with a label mentioning the name of the particular data. In both the cases we have the same symbol. So, you can use either of these set of symbols to represent different components of a DFT. **(Refer Slide Time: 12:16)**



Another important thing that we learned is about the levels. Remember that, we emphasized the point that design should be modular and hierarchical. So, that it is easier to create and understand. So, DFT notations allow us to represent a system design in a hierarchical manner as we have seen earlier. So, there should be at least 3 levels in the hierarchy. Level 0 is called a context diagram level 1 is called overview diagram and level 2 is called detail diagram.

Now, there can be levels 3, 4, 5 any number as we want all will be detailed diagrams. Too many levels of course are not good and too fewer also not good. So, we have to maintain some balance in the number of levels, that we are going for with that basic background and recap. Let us, now turn our attention to the case that is the calendar app and the design of the app system and expressing it using DFD.

(Refer Slide Time: 13:37)

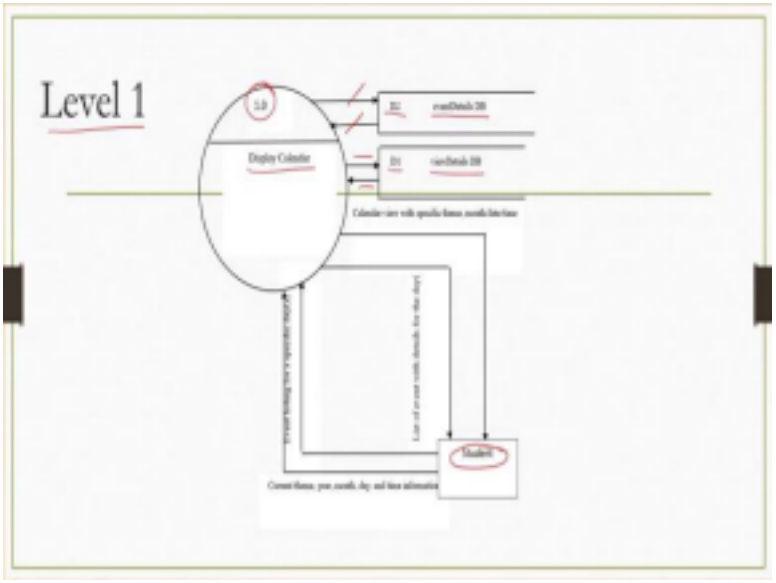


So, we will start with level 0 or the context level diagram for the calendar app as shown in this figure. Now here, we will be using the second set of symbols that we have shown earlier with some modifications for is in explaining the concept. So, we will be using the level number here, along with the name of the process, separated by a horizontal line within a circle. Although ideally, we should use an ellipse, this is just to simplify the explanation.

So, at the context level as we have seen earlier there will be only one process. Now, this process encapsulates the whole system everything in the system and the process interacts with an external entity. In this case, it is a student, who is the primary user of the app and there is data flow between the entity and the process. Some goes from the entity to the process and some comes from the process to the entity.

So, student can provide to the system data such as email account details, theme details, calendar view details, reminder details, or update set delete events date, time, information; and so on. Similarly, system can send to the student information such as synchronization success or error messages specific theme on display requested calendar view and some are update, set, delete events, date and time status. So, these are the things that can be sent to the student by the process and the process can in turn take input from the students. So, that is about level 0 or context level diagram.

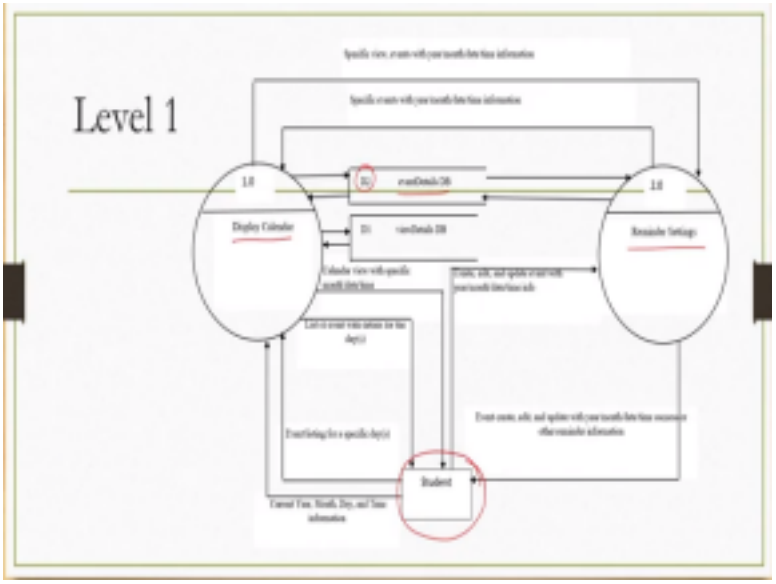
(Refer Slide Time: 16:07)



Then we come to level 1, which is an overview level. In this level, we break down the single process that is there in context level representing the whole system and here, we show the top level modules and their connection with each other. Along with data stores if they are there at this level. So, in level 1, we can see that we can have a process display calendar for the app. So, you have leveled it as 1.0, which is level 1 process. There are 2 data stores D1 and D2. D1 is about view details database and D2 is about event details database.

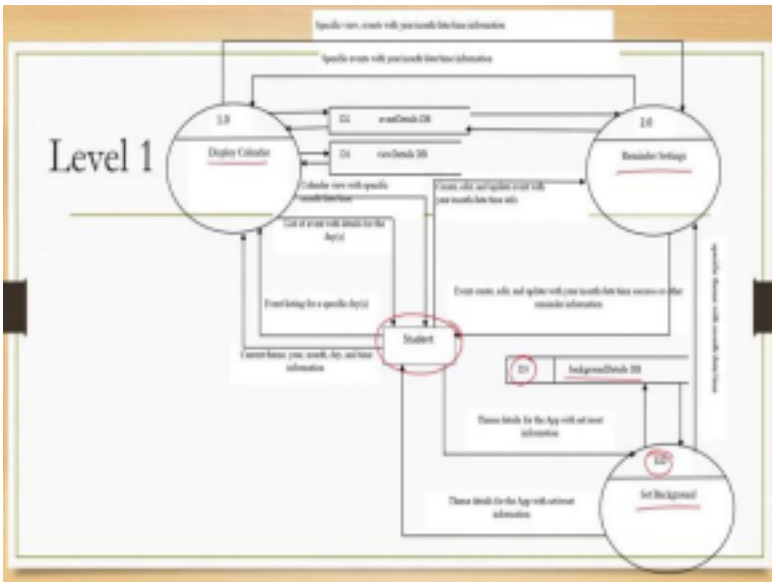
Then we have the external entity shown here student and all the data flow that we have seen earlier, with addition of data flow from data store to the process as shown here. From each data store to the process there is to and fro data flow. That is in addition to the data flow between the process and the external entity. But this is not the only process in level 1. Let us see what other top-level process can be there.

(Refer Slide Time: 17:36)



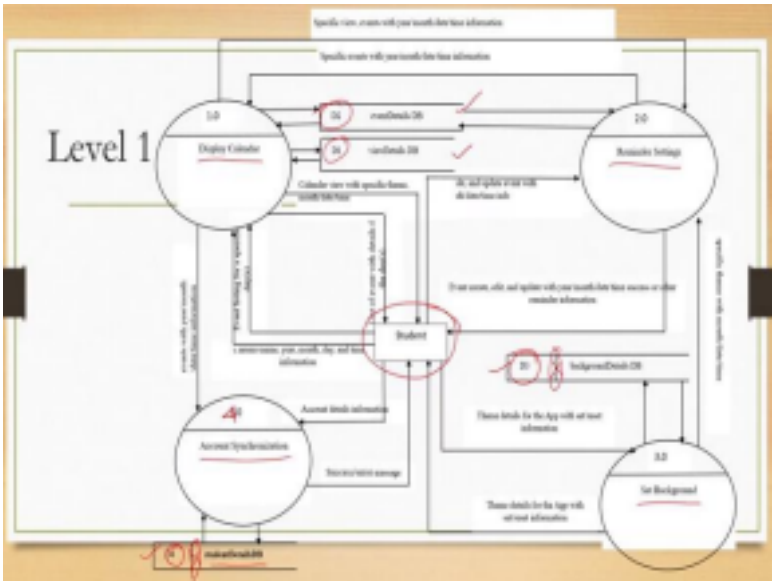
So, earlier we saw Display calendar process. Now, there can be another reminder settings process at the same level 1. Now, this process interacts with D2 event details database. But not with D1, view details database. As well as, it interacts with the external entity and the other process as shown in this figure, are these the only 2 let us see

(Refer Slide Time: 18:16)



We can have another process. So, we have already seen display calendar reminder settings. We can have a third top level process that is set background. Now, this process makes use of another data store. D3 background details and it interacts with the data store. In addition to its interaction with the external entity student in the form of data flow from and to the external entity it also interacts with the other processes particularly processed 2 that is reminder setting.

(Refer Slide Time: 19:09)



We can even add one more process it should be numbered 4 this is 4 accounts synchronization. In addition to the other three processes display calendar, reminder settings, and set background. Now, this process makes use of yet another data store which is the D4 student details database. So, there are 4 data stores then view details, event details, background details, and student details.

Now, one thing you can note here is that these 2 data stores D1 and D2 have used one notation and the other 2 data stores D3 and D4 have used another notation, where we are using the vertical bar to separate the identifier and name of the data stores this is another type of notation. So, we should never mix these things. So ideally, we should use either of these. This is just to show you that this mixing is not a good practice.

So, in this case, if we are following D1 and D2 data store conventions. Then this bar should not be there. So, in level 1 as you can see, we have now, 4 processes these are top-level processes. There are 4 data stores and there is the data flow between the processes, between the processes and data stores, and between the processes and the external entity, that is the student. So, that is about level 1 or overview level. Now, we move to level 2 or the detailed level. In this level, we detailed the individual processes that we have shown in level 1.

(Refer Slide Time: 21:18)

So, we start with process one of level one. So, process 1 is a display calendar. Now, we are showing the detailed design for this process in level 2. As you can see, here we can have 2 sub processes managing calendar view and managing of events. One is given the identifier 1.1, another one is given the identifier 1.2 and they make use of the 2 databases D1 and D2, the view details database and the event details database.

Note that here, we will be using the terms databases and data stores interchangeably, they refer to the more or less same concept from our point of view. Now, these 2 processes are interacting with the external entity student as well as the data stores. 1.2 interacts with data store 2, 1.1 interacts with data store 1 and there are inputs coming from the second top-level process P 2.0 and output going to the second top-level process P 2.0 that is process 1 level 2. **(Refer Slide Time: 22:34)**

Now, let us move to the level 2 diagrams for process 2 that is a reminder setting. Here, we have 3 sub processes or sub-functions 2.1 create events reminder, 2.2 update events reminder, and 2.3 remove events reminder. Now, 2.1 and 2.3 interact with the data stores same data store event details database. For Simplicity, we have replicated the data store notation here. Although that is not required, we could have simply used some arrows from this data store to 2.3.

As usual, process 2.1 produces an output that goes to process 1 and takes input from process 3. Process 2.2 produces an output that goes to process 1. Process 2.3 produces an output that goes to process 1. The 3 sub-processors interact with the external entity as well as the data stores D2.
(Refer Slide Time: 23:50)

Now, let us check the level 2 diagram of process 3 that is a set background. It has got 2 sub processes 3.1 set theme and 3.2 reset theme. They make use of the data stores D1 and D2. So, like before they also interact with the external entity as well as the data stores. Process 3.1 produces some output which goes to process 1 and process 2 that is process 3.0 level 2.

(Refer Slide Time: 24:34)

Finally, the level 2 diagram for process 4 is synchronization. Here we have 2 sub-processors 4.1, add and Link user email account with calendar activity. 4.2 remove or D-Link or edit email account with calendar activity. In fact, we can break it up further, but for simplicity we just kept it restricted to 2 sub-processes. Here, we make use of the D3 student details database.

Student external entity is present here. So, process 4.1, as well as 4.2, interacts with the external entity as you can see here through these data flow notations. They also interact with the data store and some input is coming to process 4.1 from process 1. So, that is level 2 detailed design of process 4. So, we have seen how we can convert the design idea namely the system design idea into a DFD using the notations that we have learned in the previous lecture.

(Refer Slide Time: 26:07)

Now, one thing here you should always keep in mind is that what we have just seen is only one out of many possibilities. The same SRS which is the starting point of our system design process can be converted to different designs. So, there is no unique solution to the design problem and we can come up with different designs. Right now, we will not go into the comparative study of which design is good, and which is bad.

The design that you are likely to choose depends on your expertise and your skill as a designer. So, if a skilled designer is there, then of course he or she is capable of coming up with the most efficient design given the set of alternatives, where the efficiency is measured in terms of manageability, resource utilization, and such considerations. But the key thing that you should remember is that from the SRS we can get many designs, what we have seen is not the only possible design there can be many other possibilities.

We have seen only one of many such alternative designs whether that is a good design or a bad design. We will not enter into that argument. Here the other component that we have learned is the entity relationship diagram or ER diagram. What is the ER diagram? Let us quickly recap and then see how we can use this knowledge to represent the data stores that we have used in our DF diagram data flow diagram for the Calendar app.

(Refer Slide Time: 27:56)

So, ER diagrams are generally used to represent data stores that we use in DF diagrams. So, they are used to express the rich internal structure organization and relationships in the data stores.
(Refer Slide Time: 28:15)

What are the basic components? There are 3 basic components one is an entity, represented with a rectangle. This is an identifiable object or concept of significance. Then we have attributes, represented within the ellipse elliptic curve is a property of an entity or relationship. And finally, the concept of the relationship which is represented with this symbol. A relationship refers to an association between entities some connection which is represented with this symbol. **(Refer Slide Time: 28:58)**

So, with these notations, we try to model a data store in terms of a collection of entities and the relationship among the entities.

(Refer Slide Time: 29:12)

So, with that basic idea let us now quickly see, how we can represent the data stores that we have just used in our data flow diagram using the notations of an ER diagram. So, we have several data stores those we can represent as student entities, calendar view entities, calendar theme entities, event entities, and reminder entities. So, these are all entities, that we can think off for the data stores that we have used in our data flow diagram.

Now, the student entity has got some attributes like name, student ID, and email student. The calendar view entity has got some attributes like view date attribute, view id, and view time. View

date is a multi-valued attribute having values such as year, month and date. Similarly, view time is a multi-valued attribute with values that are our minute and second. Calendar theme the other entity has got 3 attributes ID, background, and color.

Now, the calendar theme, calendar view, and student these three entities are related to each other with the relationship get the student can get calendar view or calendar theme. So, in that way, they are related. So, this is not a binary relationship. If you recollect it, is a ternary relationship. Then, there is this event entity and reminder entity. The event entity has got several attributes such as event end time, event date, event ID, description type, start time, and name of the event.

The reminder entity has got several attributes scheduled time, description ID, reminder ID, as well as the event for which remainder it is said to be set that ID. Now, event and reminder entities are connected through his relationship. So, event as reminders and reminders as an event so, in both ways it is applicable. Similarly, the event entity and the student entity are connected through a relationship which is editing an assignment relationship.

So, students can edit events and students can assign events or events can be assigned to students. So, both ways are applicable. So, both the arrows are there. So, in this way, we can model the data stores, that we have used in the data flow diagram. One thing that may be noted here is that the data store levels or the rather names are not exactly the same as the names that we are using to represent entities' relationships and attributes.

So, that mismatch may be there however overall, the kind of data that we are using in the data flow diagram is our main focus, and that data we can model using the ER diagram notations. Even if, we need to use different levels for representing entities or relationships or attributes. So, in this example, you can also see the idea of ternary relationships as well as multi-valued attributes.

So with that, we have come to the end of this lecture. So, in this lecture we tried to learn the idea of DFD and ERD in a better way in terms of one case study, where the case is the calendar app that we are using throughout our course. We have seen how to come up with a level 0 diagram for the app level 1 or an overview diagram for the app, as well as level 2 or detail diagrams for each of the top-level modules or the processes that are part of the level 1 diagram.

Several data stores we have used in the DFD which we can model and represent using ER diagram notations which we have also seen in this lecture that is all for this lecture. I hope you enjoyed the material and you got a better understanding of the concepts of DFD and ERD and how to use them in practice? In the next lectures, we will take up another way to go for system design which is the object-oriented design approach looking forward to meet you all in the next lecture thank you and good bye.