

**Software Engineering**  
**Prof. Rajib Mall**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**

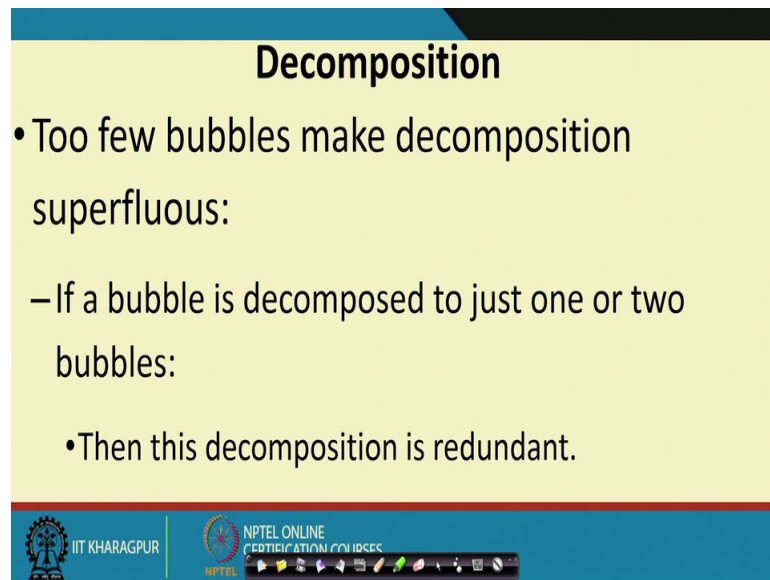
**Lecture – 25**  
**Developing DFD Model**

Welcome to this lecture. In the last lecture we were discussing how to develop the DFD model of any problem. We saw that developing the context diagram is extremely simple. We just draw a circle, write the name of the system in it, identify the external users or the external entities and see what type of data they produce they input to the system and what type of data they consume. And that forms our context level diagram or the level 0 diagram and then for the level 1 diagram we just take up the SRS document and identify the functional requirements. If there is anything between 3 to 5 functions then you directly represent them on the DFD diagram level 1 diagram.

But if there are large number of functions in the SRS document very sophisticated system then we need to group some of the functions in the higher-level functions. And normally in a good SRS document that is already done that the functional requirement consists of subsections and each subsection contains several functionalities. So, we can represent those subsections in the level 1 diagram and then we identify the data flows that occur among these functions.

And then we do the decomposition to develop the still higher-level diagrams that is more detailed diagrams. And for decomposition we need to take one bubble at a time and then see what are the sub functions and we need to decompose between 3 to 5 sub functions in the next level diagram. And we were discussing that the; if there are two or few bubbles in the next level then the decomposition does not achieve anything.

(Refer Slide Time: 02:38)



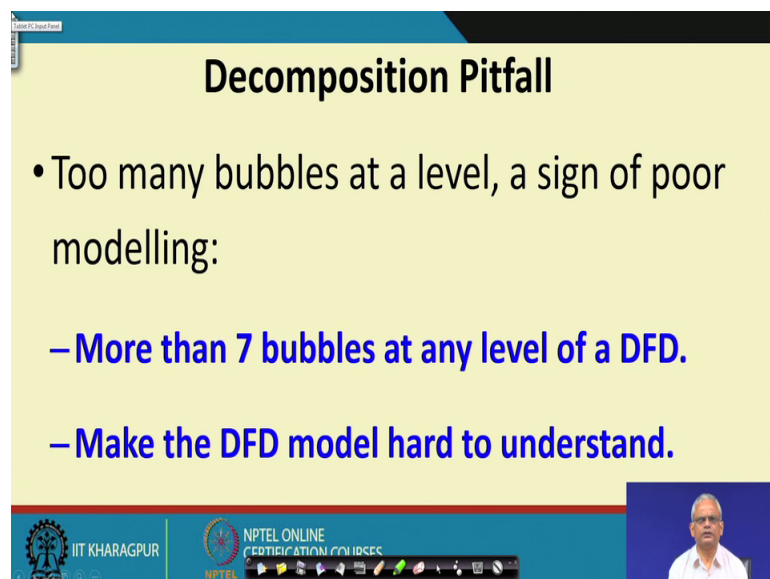
### Decomposition

- Too few bubbles make decomposition superfluous:
  - If a bubble is decomposed to just one or two bubbles:
    - Then this decomposition is redundant.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Let us say if there was one bubble in some diagram, some DFD and in the next DFD again we give one bubble then there is no decomposition. And even if we decompose one bubble into two bubble then that is also not very meaningful. Typically, we need to decompose each bubble into 3,5 or up to 7 bubbles. So, we will not be decomposed into too many bubbles because that will make difficult to understand. So, the hierarchical nature will be lost and each level will become too much of details and it will become difficult to understand.


(Refer Slide Time: 03:32)



### Decomposition Pitfall

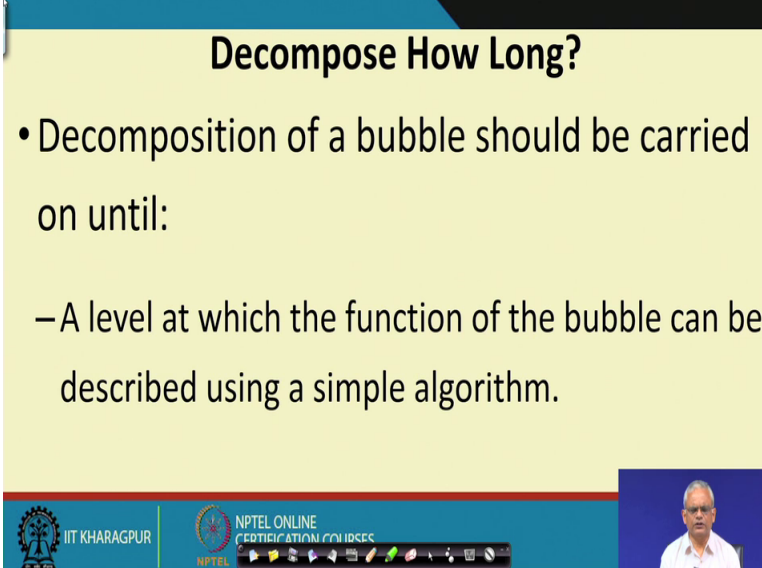
- Too many bubbles at a level, a sign of poor modelling:
  - **More than 7 bubbles at any level of a DFD.**
  - **Make the DFD model hard to understand.**

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES



So, at any level or any DFD diagram at any level we should have about 7 maximum bubbles and that will be a good DFD diagram. It will be easy to understand, otherwise if there are too many bubbles then the diagram would become difficult for somebody to understand.

(Refer Slide Time: 04:01)



**Decompose How Long?**

- Decomposition of a bubble should be carried on until:
  - A level at which the function of the bubble can be described using a simple algorithm.













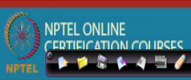

The slide features the IIT Kharagpur logo on the left, the NPTEL Online Certification Courses logo in the center, and a small video inset of a speaker on the right.

We then keep on decomposing each bubble take up one bubble in a diagram, decompose and then again take up those known de decomposed diagram one bubble and go to decompose it in the next level and so on. But to what extent we go on decomposing? We stop our decomposition when we find that it is decomposed to a function, which can be described using a very simple algorithm and its coding should not take more than a few statements.

(Refer Slide Time: 04:44)

### Example 1: RMS Calculating Software

- Consider a software called RMS calculating software:
  - Reads three integers in the range of -1000 and +1000
  - Finds out the root mean square (rms) of the three input numbers
  - Displays the result.













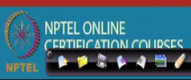
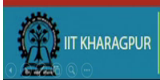



Now, let us do one example problem and normally when we learn anything we start doing something very simple and then we do more sophisticated problems. The simplest problem that we do is called as the RMS calculating software. It is a rather trivial software which just reads a number reads three numbers between minus 1000 and plus 1000, computes their root mean square or RMS and then displays the result, very very simple software.

(Refer Slide Time: 05:30)

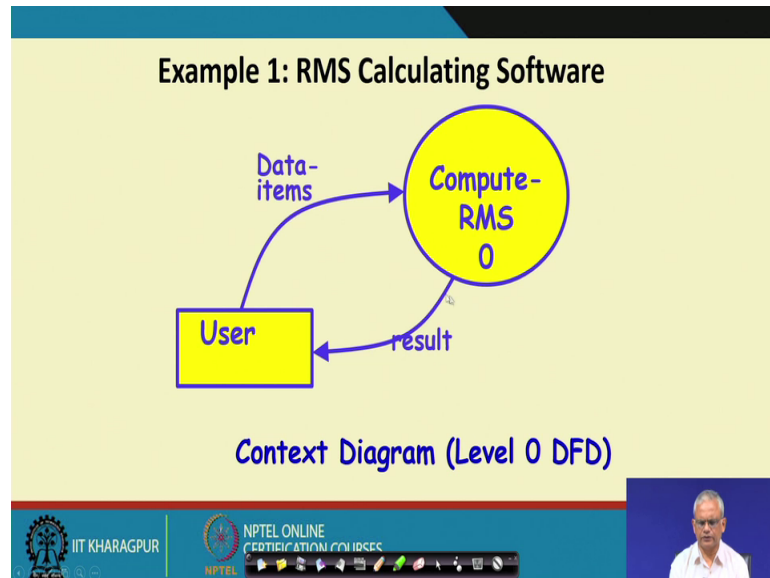
### Example 1: RMS Calculating Software

- The context diagram is simple to develop:
  - The system accepts 3 integers from the user
  - Returns the result to him.



Now, let us try to do the context diagram. In the context diagram as you said we need to just draw a circle and represent the system and we write the name of the system RMS software. Only at the context level a noun form of the process is allowed, in the other levels we typically give a verb form to the process. And then it accepts 3 integers and produces the result and we have the user for this.

(Refer Slide Time: 06:29)



So, the same thing we have written it here in this slide. The name of the software is compute RMS and then there is this user who enters data items and the data items are 3 integers and it produces a result. So, this is our context diagram or the level 0 DFD; it is extremely simple to draw.

(Refer Slide Time: 07:00).

**Example 1: RMS Calculating Software**

- From a cursory analysis of the problem description:
  - We can see that the system needs to perform several things.

Now, this is a very simple problem it just consists of a one function. Any decomposition of this into level 1, level 2 etcetera will become highly artificial because this is a very simple problem.

(Refer Slide Time: 07:18)

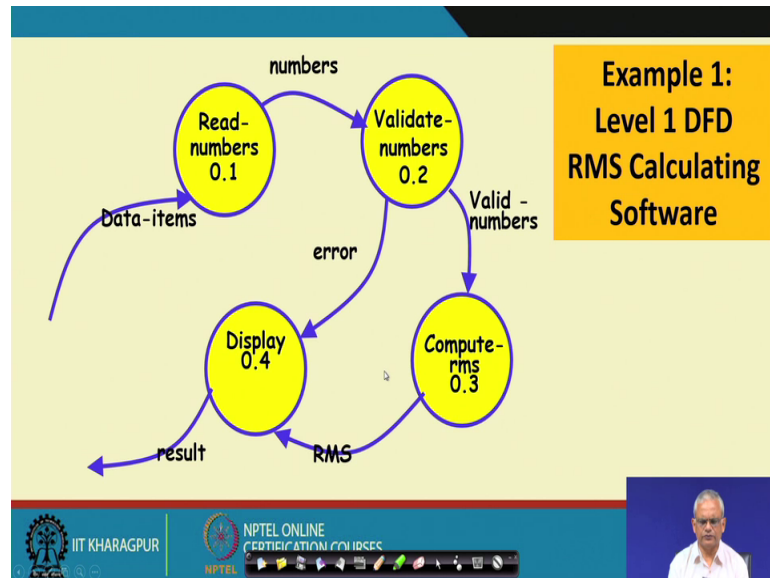
**Example 1: RMS Calculating Software**

- Accept input numbers from the user:
- Validate the numbers,
- Calculate the root mean square of the input numbers
- Display the result.

But then if we really want to decompose we will see that for performing the RMS calculation, it first needs to read the numbers from the user; then do some validation that they are within minus 1000 to plus 1000. And then perform the computation root mean square and then display the result. So, these are the sub functions of the level 1 sorry

level 0 context diagram because there is only one functionality, it is a very simple software. So, in the level 1 we decompose the function represented in the level 0 and then we can represented in this form.

(Refer Slide Time: 08:00)



That if read numbers, read the data items that are entered by the user just look at here that in the level 1 diagram we do not represent the external entities. The external entities are represented only in the context diagram or the level 0 diagram. The data items are entered into the read numbers, produces the validated numbers sorry it reads the numbers and inputs the number to validate numbers. It validates and then gives the valid numbers to this compute a RMS otherwise, it produces a error message. The error message is given to the display a bubble otherwise, it may give an RMS message that is the result and it produces that display, it may be an error or it may be the RMS.

So, that is the level 1 diagram, but as I was saying that there is a highly simple problem and therefore, a level 1 diagram becomes a very superfluous and very artificial. Nobody would really do a decomposition of a compute RMS that is a very simple function by itself. And normally we would not decompose that and in real problem modeling DFD modeling we do not decompose to this level, but just to give an illustration how decomposition can be done I explained that with the simplest example.

(Refer Slide Time: 09:52)

**Example: RMS Calculating Software**

- Decomposition is never carried on up to basic instruction level:
  - A bubble is not decomposed any further:
    - If it can be represented by a simple set of instructions.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, I just mentioned here that we do not really decompose to this level. Compute RMS itself is a very simple function and that can be represent, that can be coded using only very few instructions.

(Refer Slide Time: 10:11)

**Data Dictionary**

- A DFD is always accompanied by a data dictionary.
- A data dictionary lists all data items appearing in a DFD:
  - Definition of all composite data items in terms of their component data items.
  - All data names along with the purpose of the data items.
- For example, a data dictionary entry may be:
  - $grossPay = regularPay + overtimePay$

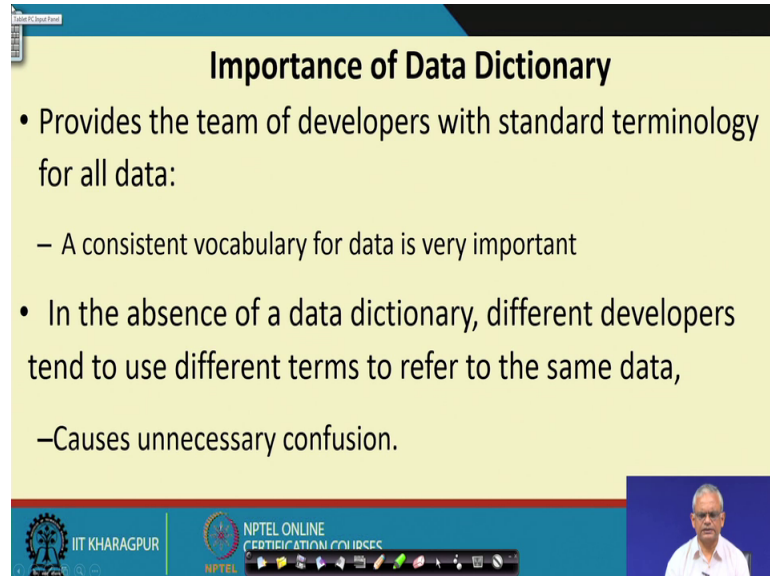
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Whenever, we do a DFD model a data dictionary must accompany it. The data dictionary lists all the data items that appear in the DFD and then write the meaning of that data item. And sometimes we might have to write a data item in terms of its component data.



So let me repeat that, that whenever we have a DFD model we must have a data dictionary accompanying that.

(Refer Slide Time: 11:38)



**Importance of Data Dictionary**

- Provides the team of developers with standard terminology for all data:
  - A consistent vocabulary for data is very important
- In the absence of a data dictionary, different developers tend to use different terms to refer to the same data,
  - Causes unnecessary confusion.

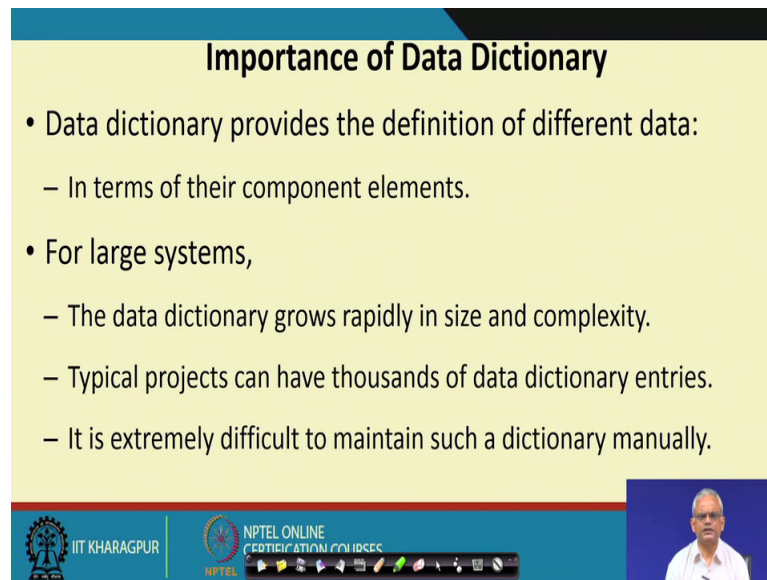
IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

The data dictionary gives the meaning of all the data items that appear on the DFD. As I saying that a DFD model can consists of a large number of DFDs and all the data names appearing in all these various DFSs are listed in the data dictionary and their meaning is given. If it is a composite data item, it is expressed in terms of the simpler data items, but for the simplest data item we just write its purpose. So, that by looking at the data dictionary you should be clear about what are the meaning of the data that appear on various DFDs.

The data dictionary is a very valuable document that is produced during structured analysis. The team members refer to the various data using the name and the decomposition given in this the data dictionary. If we do not have a data dictionary, there is a chance of misuse. People may the team members may misunderstand the data that is represented on the DFD and also while they refer to different data items, they may use inconsistent names and that will make it very confusing.

So, the data dictionary standardizes the names of the data and explains their purpose and what they consist of. The complex data items consists of maybe some simpler data it mentions them.

(Refer Slide Time: 12:41)



**Importance of Data Dictionary**

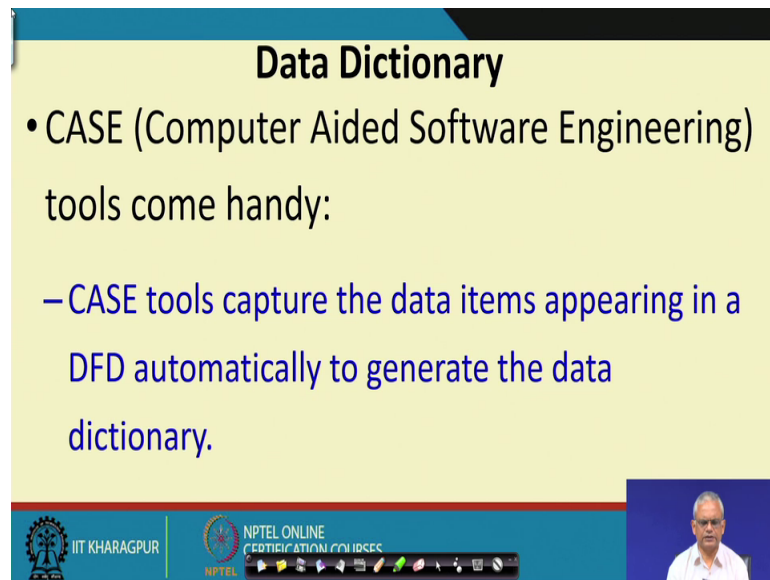
- Data dictionary provides the definition of different data:
  - In terms of their component elements.
- For large systems,
  - The data dictionary grows rapidly in size and complexity.
  - Typical projects can have thousands of data dictionary entries.
  - It is extremely difficult to maintain such a dictionary manually.

IIT KHARAGPUR | NPTEL ONLINE EDUCATIONAL COURSES

For very simple systems, the data dictionary is quite manageable. If you have 10 or 20 data items then you can just write down on a piece of paper or something and that will serve the purpose. But then in real projects the data dictionary can sub tens of thousands of entries and if you start to manually write this then it becomes very difficult.

You may do mistakes or there can be redundant entries and when you want to search the name of some data item becomes very difficult to search it manually. And therefore, all DFD tools, structured analysis tools they capture the data that appears in the DFD and build the data dictionary and maintain it in a relational database so, that query etcetera becomes easy.

(Refer Slide Time: 13:55)



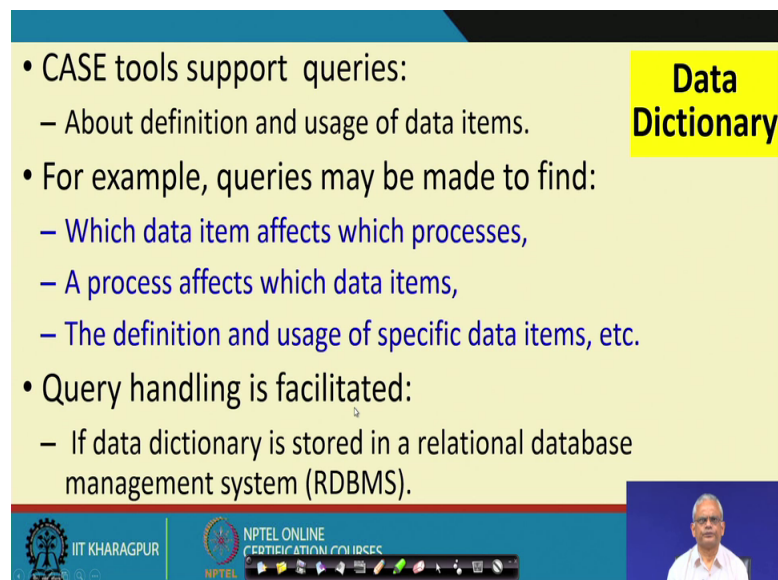
**Data Dictionary**

- CASE (Computer Aided Software Engineering) tools come handy:
  - CASE tools capture the data items appearing in a DFD automatically to generate the data dictionary.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, all case tools help maintain the DFDs automatically and they help to automatically search the data items.

(Refer Slide Time: 14:12)



**Data Dictionary**

- CASE tools support queries:
  - About definition and usage of data items.
- For example, queries may be made to find:
  - Which data item affects which processes,
  - A process affects which data items,
  - The definition and usage of specific data items, etc.
- Query handling is facilitated:
  - If data dictionary is stored in a relational database management system (RDBMS).

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

(Refer Slide Time: 14:22)

• Composite data are defined in terms of primitive data items using simple operators:

- **+**: denotes composition of data items, e.g.
  - **a+b** represents data a together with b.
- **[,]**: represents selection,
  - Any one of the data items listed inside the square bracket can occur.
  - For example, **[a,b]** represents either a occurs or b

**Data Definition**

$C = a + b$

IIT KHARAGPUR | NPTEL ONLINE EDUCATIONAL COURSES

The case tools we will see that we they can also answer query like a which data item is affecting which processes. A process will affect which data items whereas, specific data item is used where it is assigned values and so on. It becomes easy to answer this kind of question when these are maintained in a automated tool using a relational database system.

Now, let us see the grammar by which these primitive items are combined into more complex data items. The rules are very simple, the symbols here. If you write a plus b then we mean that a together with b and if we write c is equal to a plus b then c consists of a and b. If we use this square bracket it means option, we write a comma b in square bracket it means that either a or b.

(Refer Slide Time: 15:59)

**Data Definition**

- **( )**: contents inside the bracket represent optional data
  - which may or may not appear.
  - **a+(b)** represents either a or a+b
- **{ }**: represents iterative data definition,
  - **{name}5** represents five name data.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

If we write a bracket b then it represents that either a or a and b. If it is in curly bracket then it represents the repetition. If we write name 5 that means, 5 data items 5 name data items.

(Refer Slide Time: 16:26)

**Data Definition**

- **{name}\*** represents
  - zero or more instances of name data.
- **=** represents equivalence,
  - e.g. **a=b+c** means that a represents b and c.
- **\* \***: Anything appearing within \* \* is considered as comment.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

If we write name star it represents many name data items; equal to is of course, equivalence. So, a equal to b plus c means a consists of b and c. So, a is a composite data item containing b and c. As you can see that the data definition is very simple using few

operators like plus, square bracket, curly bracket and so on. And we can also write comments here within two stars.

(Refer Slide Time: 17:09)

• numbers=valid-numbers=a+b+c

• a:integer \* input number \*

• b:integer \* input number \*

• c:integer \* input number \*

• asq:integer

• bsq:integer

• csq:integer

• squared-sum: integer

• Result=[RMS,error]

• RMS: integer \* root mean square value\*

• error:string \* error message\*

**Data Dictionary for  
RMS Software**

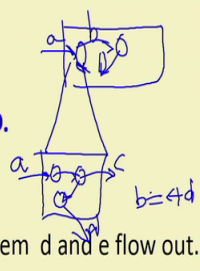
IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

This is just one example here, a is a primitive data and I have written a is a integer. Numbers is a composite data which is also a synonym is valid numbers and it consists of three components a, b, c. Result is a data which sometimes can be RMS and other times it can be error and RMS and error are primitive data item. And we have also mentioned here in comment, the purpose for which that used; error is used to represent a error message, RMS is used to represent the root mean square value.

(Refer Slide Time: 17:59)

### Balancing a DFD

- Data flowing into or out of a bubble:
  - Must match the data flows at the next level of DFD.
- In the level 1 of the DFD,
  - Data item c flows into the bubble P3 and the data item d and e flow out.
- In the next level, bubble P3 is decomposed.
  - The decomposition is balanced as data item c flows into the level 2 diagram and d and e flow out.



The diagram illustrates the concept of balancing a Data Flow Diagram (DFD). It shows two levels of decomposition. In the first level, a process bubble labeled 'P3' has an input data flow 'c' and two output data flows 'd' and 'e'. In the second level, bubble 'P3' is decomposed into three smaller bubbles. The input 'c' from the first level flows into the top bubble, which produces output 'd'. The middle bubble produces output 'e'. The bottom bubble produces output 'd' and 'e'. The equation  $b = c + d$  is written next to the bottom bubble, indicating that the total output of the decomposition (d + e) must equal the input (c) of the parent bubble.

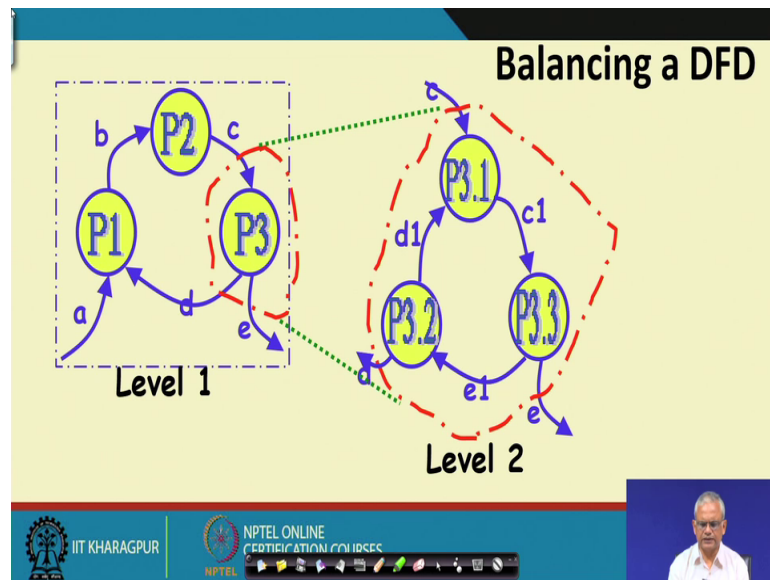
IIT KHARAGPUR | NPTEL ONLINE EDUCATIONAL COURSES

Now, if we practice a few examples few more examples we will see that given an new example you can easily draw the DFD model. But, before we do that lets look at another important concept in the DFD modeling called as balancing a DFD. Here we check, if we decompose a DFD into a lower level DFD then the data that is input that is flowing in and out of the DFD matches the next level. What we mean here is that let us say we have a DFD model and we have some DFDs here.

Now, let us say we decompose this bubble take a 1 bubble here and we decompose into a DFD diagram. So, we again let us say decompose into 3 bubbles these diagram and it takes these data produces a data and this lets say finally, produces a data and this also produces a data. Now, let us say this takes a and produces b; then in the next level we must also have a taken in here and this let us say produces c and d, we must have b is equal to c plus d.

This is balanced because this bubble here the input is a produces output is b and when you have decomposed it into the next level, the data that is flowing into this diagram is b just coming out from the diagram is c and d. And for balancing we see that the input is balanced and for the output to be balanced c plus d must be equal to b. So, this rule of balancing is used it all decomposition levels.

(Refer Slide Time: 20:42)



This is another example here balancing. Let us say we have a level 1 DFD representation, it is this that P 1 P 2 P 3 at 3 processes and they are exchanging some data. And a is consumed by P 1 P 3 consumes c and e sorry the input to P 3 is c and P 3 produces d and e. Now, let us say in the next level we were decomposing this bubble here. So, this is the next level diagram again decompose into 3 bubbles, there are some data that are internally input across these different bubbles.

But just see here that to this bubble c is input and this bubble our outputs d and this outputs e. Now, we check here yes c is input here, d and e are output here and therefore, it is a balanced decomposition. But let us say we had another input here then that would not be balanced.



(Refer Slide Time: 22:17)

• Number the bubbles in a DFD:

- Numbers help in uniquely identifying any bubble from its bubble number.

• The bubble at context level:

- Assigned number 0.

• Bubbles at level 1:

- Numbered 0.1, 0.2, 0.3, etc.

• When a bubble numbered x is decomposed,

- Its children bubble are numbered x.1, x.2, x.3, etc.

**Numbering of Bubbles**

0.1.2

IIT KHARAGPUR | NPTEL ONLINE EDUCATIONAL COURSES

Another concept is numbering DFD. So, far we saw that we just write to write the name of the process in the bubble, but if we write a number that will help us in identifying the bubble. For example if we write 0 is only for the context level, if we write the bubble 0 we can easily find out which bubble it is.

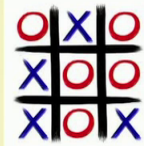
In the next level, if we in the level 1 we write 0.1 0.2 then by just looking at the number we would know which diagram and that is the reason why the bubbles are normally numbered; both by the tools and even if we are doing it manually. The context level is assigned the number 0 that is the convention and at level 1 we decompose into a set of bubbles. Let us say we decompose into 3 bubbles, then we number them 0.1 0.2 0.3 meaning that the parent of this bubble is 0 that is context.

And similarly, if we let us say find a bubble whose number is 0.2.1 0.2.1 and then we know that the level 1 diagram the second bubble, this is the parent of this and for that the parent is the 0. So, this is a level 1 diagram sorry level 2 diagram and so, 0.1.2. So, this is a level 2 diagram the parent of this diagram is 1 and for that this is the context diagram.

(Refer Slide Time: 24:44)

**Example 2: Tic-Tac-Toe Computer Game**

- A human player and the computer make alternate moves on a 3 X 3 square.
- A move consists of marking a previously unmarked square.
- The user inputs a number between 1 and 9 to mark a square
- Whoever is first to place three consecutive marks along a straight line (i.e., along a row, column, or diagonal) on the square wins.



IIT KHARAGPUR | NPTEL ONLINE EDUCATIONAL COURSES




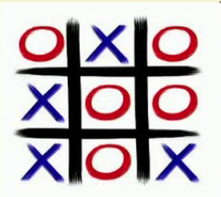
Now, let us do one more problem which is the Tic-Tac-Toe game. Let us first understand the Tic-Tac-Toe game, I think most are familiar with this kind of game. Here it is played between the computer and the user. Let us say the player marks let us say by this red 0 and the computer makes let us say cross. So, the player mark is square here and then the computer will mark something, in response to that the user will mark another square and so on.

The player wins if we can occupy a three consecutive elements in the row or column or diagonal. So, before the player can mark all along this diagonal the computer has occupied the this one so, that the player cannot win straight away. And if the game all the squares are over and no one has got any mark along a row or a column or a diagonal then the game is considered draw.

(Refer Slide Time: 26:30)

**Example: Tic-Tac-Toe Computer Game**

- As soon as either of the human player or the computer wins,
  - A message announcing the winner should be displayed.
- If neither player manages to get three consecutive marks along a straight line,
  - And all the squares on the board are filled up,
  - Then the game is drawn.
- The computer always tries to win a game.



So, there are 9 places to fill and each player that is the human player or the computer they place their marks here. Now, can we draw the context level diagram of this? The game is intuitively clear and we writing the software for this and we want to draw a context level diagram for this problem. Please try this and we will discuss the solution in the next class. And as I was mentioning that if we can do few problems, we can develop the DFD model then given even a very sophisticated problem we can easily develop its DFD model.

We will stop now; continue from this point in the next class.

Thank you.