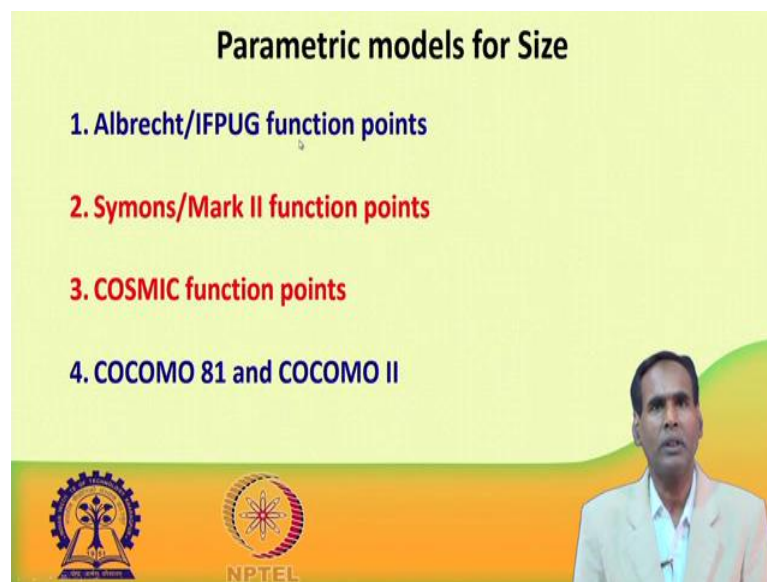


Software Project Management
Prof. Durga Prasad Mohapatra
Department of Computer Science and Engineering
National Institute of Technology, Rourkela

Lecture – 20
Project Estimation Techniques. (Contd.)

Good afternoon. Now let us start in this section, in this lecture we will take up some of the examples for this Albrecht a function point methods. We will take few examples so, that you can better understand. We will take some more examples on this Albrecht IFPUG function points then, we will take two more types of function points that is Symons Mark II function point and the COSMIC function points.

(Refer Slide Time: 00:43)



So, we have already discussed this parametric models for size, last class I have told we will use a we will see 4 types of this a estimations like IFPUG function points, Mark II function points, COSMIC function points and COCOMO 81 and COCOMO II. Last class already we have discussed IFPUG function points Now, we will see this Symons Mark II function point and COSMIC function points, before going to that let us take one or two more problems more examples on this IFPUG function points.


(Refer Slide Time: 01:17)

Example on Albrecht/IFPUG FP

Consider a project with the following functional units:

- 50 user inputs, 40 user outputs, 35 user enquiries, 6 user files, 4 external interfaces. Assume all complexity adjustment factors and weighting factors are average.
- Compute the function points for the project.
- Suppose that program needs 70 LOC per FP.
- Find out the size of complete project.

*UFP = \sum (no. of elements of given type) * (wt.)*



So, we will take up this example, but I hope that before going to this example let me again tell you let us take up that this example. And here, we will see that how this previous formula that we can use I have already told you last class, first we have to find out the unadjusted function point, then we have to refine it by using the value adjustment factor. So, this formula for unadjusted function point can be written as what? I have already shown it in the last slides, this will be equal to summation of number of elements of the given type maybe it is a input or output or a query, what is the number of elements into the weight



The weight I have already given you for simple there can be three; the weights can be divided into three categories depending on whether it is simple or medium or high. So, that table Albrecht might that complexity classifiers the multipliers we have already given you in the last class. And now, with this now let us take up this problem ok. So now, let us take this one more example, this example say that consider a project with the following functional units there are suppose in that project there are 50 user inputs, 40 user outputs, 35 user enquiries and 6 user files, 4 external interfaces.

So, now let us assume that all the complexity adjustment factors and weighting factors they are average, everything is average. So, then the question is that compute the function points for the project and then compute what will the size? Given that the program needs 70 LOC per function point.

(Refer Slide Time: 03:29)

Solution

- $UFP = 50 * 4 + 40 * 5 + 35 * 4 + 6 * 10 + 4 * 7$
 $= 200 + 200 + 140 + 60 + 28 = 628$
- $VPF = 0.65 + 0.01 (14 * 3) = 1.07$
- $AFP = UFP * VPF = 628 * 1.07 = 672$
- $Size = FP * (LOC \text{ per FP}) = 672 * 70 = 47040 \text{ LOC}$




So now, let us go you can see that the inputs is equal to 50, outputs is equal to 40 and queries is 35 and internal files you can see that number of internal files is 6 and 4 is the external interfaces that 6 and this 4.

(Refer Slide Time: 03:38)

Another example

- Compute the function-point value for a project with the following information-domain characteristics.
 - Number of user Inputs: 32
 - Number of User Output: 60
 - Number of User Inquiries: 24
 - Number of Files: 8
 - Number of External interface: 2
- Assume that all complexity adjustment values are average.



So, use these values in the given formula that already I have given you and then it will give you this values of UFP. So, basically two things I am taking the possible number of elements from each category input, output, inquiry then internal file and external interfaces multiplied by the multiplier the weight, which is whether it will vary

depending on whether it is a simple or average or what high. In my previous problem I have already told you that these are all what average. So, all the complexity adjustment factors are average, as well as these weighting factors they are average, it will be much more easy to compute.

So, now UFP is equal to state formula apply this you will get 628, then we have to find out the, what value adjustment factor. Again that formula we have told you $0.65 + 0.01$ into TDI, where TDI is equal to Total Degree of Influence. And here 14 parameters since, it is told that this is average please see I have already told you that these complexity adjustment factors are average, as well as the weighting factors that mean the GSC this General System Characteristics there, these are these are also average.

So, again for average since you have taking a scale from 0 to 5, again this is average will coming 3 so, this will 1.07. So, the AFP will be the adjusted function points will be equal to unadjusted function points into VAF. So, because why we are doing this? We want to refine it. So, the you can get the adjustment function point by what multiplying the unadjusted function point on the value adjustment factor. So, this is given to 672. So, this is the adjusted function points.

So, now what is question again and the second part question is that find out the size of the complete project, how you can compute the file find out the complete the size of the complete project, you have to use this information. It said that the program needs 70 LOC per function point; that means, per function point that can be 70 lines of code. So, what is the total number of lines of code; so, sorry what is the total number of FP function point 672. So, the total size will be equal to the function points into LOC per function points. So, 672 into 70, in this way, you will get that these total size of the project in terms of LOC is coming to be 47040.

So, in this way for small projects you can easily use this formula find out the function points and size. So, in the examination some this types of small what questions might be asked and you have to find out the answer. So, let us take quickly another example in these example a project you have to develop for that you have to find out the function point for that project, this project has the following characteristics there are number of user inputs is 30, number of user outputs is 60, number of user inquiries 24, number of files is equal to 8 and number of external interface is 2.

So, and assume that all the complexity adjustment value are average and all the weighting factors are average.

(Refer Slide Time: 07:09)



The slide, titled "Solution", displays the following calculations:

- $UFP = 32 \times 4 + 60 \times 5 + 24 \times 4 + 8 \times 10 + 2 \times 7 = 128 + 300 + 96 + 80 + 14 = 618$
- $VAF = 0.65 + 0.01 \times (14 \times 3) = 0.65 + 0.42 = 1.07$
- $AFP = UFP \times VAF = 618 \times 1.07 = 661.26$

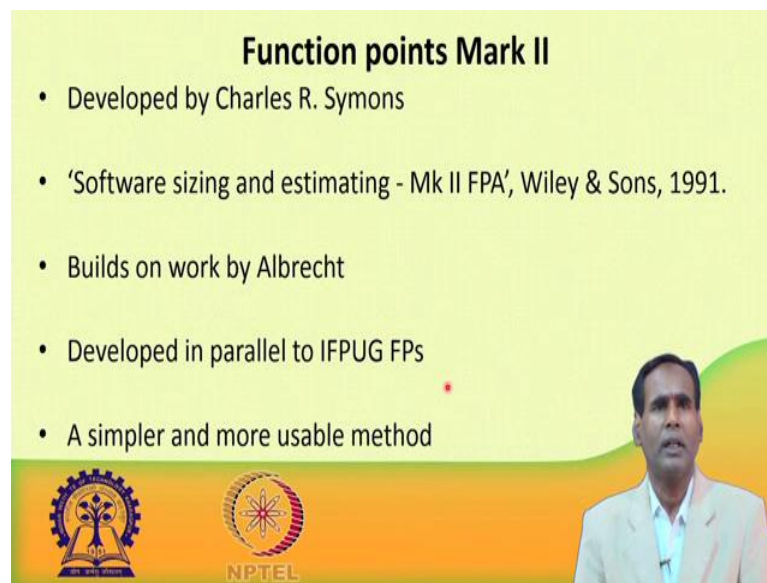
The slide also features the logos of IIT Bombay and NPTEL, and a small video inset of a man in a white shirt.

So, then you have to find out the number of function points, you can see UFP again straight forward formula that I have given summation of the number of elements from each type into weight, whole then take the overall summation you will get applying this formula straight forward. You can see that there are 32 number of what input users and we know, therefore, average case this what value is 4, it has already given these are adjustment factors are average. So, the value will be what 32 into 4. Similarly output is 60, the average weight is 5 and the inquiries it is 24 number of inquiries are there you can see, and this average value for inquiry is 4, unlike this you can compute there are the 8 numbers of what internal files and 2 number of external interfaces.

So, 8 into average value for the interfaces 10 and 2 numbers of external interfaces, average value it is 7. So, find out this value it is coming 618 then, you have to compute the value adjustment factor by using the formula 0.65 plus 0.01 into TDI and since, it is given that is these are also the general system characteristics they are average so, again 14 into 3 so, that will 1.07. So, finally, the adjusted function point can be obtained by multiplying the unadjusted function points and the value adjustment factor. So, this is giving to be 661.26. So, the total number of adjusted function points or the adjusted function point counts for this project will be equal given as 661.26.

So, suppose if for function point you have to write say 80 lines of code. So, what will the size in terms of LOC? So, 661.26 and 80 so, this will give you roughly the size of the proposed project. So, in this way you can use Albrecht function point method to find out the, to estimate the size of a given project.

(Refer Slide Time: 09:07)



Function points Mark II

- Developed by Charles R. Symons
- 'Software sizing and estimating - Mk II FPA', Wiley & Sons, 1991.
- Builds on work by Albrecht
- Developed in parallel to IFPUG FPs
- A simpler and more usable method

The slide features the IIT Bombay logo on the left and the NPTEL logo in the center. A small inset video of a man in a light-colored suit is visible in the bottom right corner of the slide.



So, now, let us quickly say about these what function points Mark II this is also another interesting approach to find out the function points. This was developed by Charles R Symons, actually the details of each method is available in a book he has developed, he has published a book on software sizing and estimating by Wiley and Sons in 1991, which contains the details of his function points, that is function points Mark II.

It is builds on; it is actually it is on extension of this Albrecht function point. So, it is build on work by Albrecht function point method, it is developed in parallel to IFPUG FPs. So, simultaneously it was developed in parallel with the Albrecht IFPUG FPs. So, it is a rather simple and more usable method.

(Refer Slide Time: 09:57)

Function points Mk II cont...

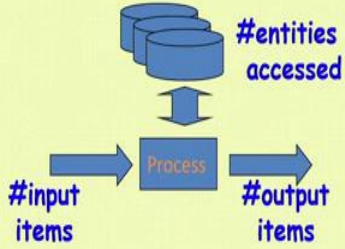
- For each transaction, count
 - data items input (N_i)
 - data items output (N_o)
 - entity types accessed (N_e)





So, how it works? So, for each transaction you have to count the data items input, the data items output and the entity types accessed. So, these three parameters you have to count and then you can find out the total function point for the whole project by using a formula, which we will see in the next slide.

(Refer Slide Time: 10:20)

Function points Mk II cont...



- Simpler than FP
- Widely used in UK

$$FP \text{ count} = N_i * 0.58 + N_e * 1.66 + N_o * 0.26$$


See how does it works, as I have already told you, you have to supply the input items as well as you will see also three parameters are required I have told the number of input

output; the number of input items supplied to the process, the number of entities assessed from this data store and the number of output items are produced.

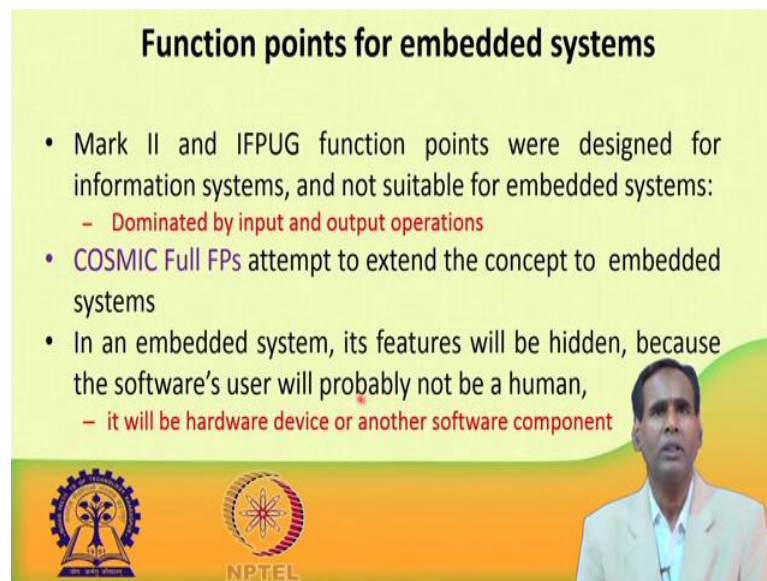
So, in this way this function point Mark II this works this is a simpler method than this Albrecht function point and it is widely used in UK. So, what is the final formula for finding out the Mark II function point, the function count is given by the following equation

$$\text{FP count} = N_i * 0.58 + N_e * 1.66 + N_o * 0.26$$

Where N_i N_e and N_o we have already defined earlier, N_i the data items input N_o is the data items output and N_e is the entity types associated or the sorry the entity types accessed.

So, this value judge; the these constants are some weights 0.58, 1.66 and 0.26, these are weights. This Simpson ok; Symons he had found out these values of the constants by performing experiments in his lab. So, directly he has utilized these values of the constants along with these parameters N_i , N_e and N_o and he has developed this formula to find out the function points, this is known as Symons Mark II function points this is simpler than Albrecht function point.

(Refer Slide Time: 11:56)



Function points for embedded systems

- Mark II and IFPUG function points were designed for information systems, and not suitable for embedded systems:
 - Dominated by input and output operations
- COSMIC Full FPs attempt to extend the concept to embedded systems
- In an embedded system, its features will be hidden, because the software's user will probably not be a human,
 - it will be hardware device or another software component

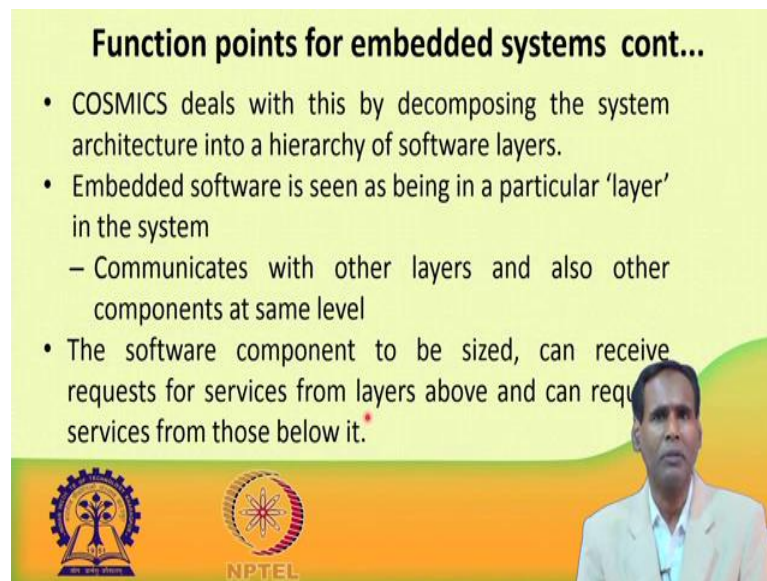
The slide features a yellow background with a green and orange gradient at the bottom. It includes the logos of IIT Bombay and NPTEL, and a small video inset of a man in a white shirt.

Now, let us see about the whether these function points so far we have seen the Albrecht function points and the Symons function points, can they be extended to real time

systems and embedded systems. See this Mark II and IFPUG function points, they were mainly designed for information systems and hence they are not suitable for embedded systems why? Because these two function points they are largely or heavily dominated by the input and output operations, the most emphasis is given on the number of input operations and the output operations. So, in order to overcome this problem so, cosmic full function points they have been developed in order to overcome these problems, this a cosmic full function points they attempt to extend the concept, this concept which concept this the Mark II concept or IFPUG concept to embedded systems.

So, Now these concepts have been extended to handle embedded systems to apply on embedded systems or real time systems. You know that in an embedded system it is features like hidden and etcetera is not it. If in an embedded system it is features normally will be hidden, because the software, the software's user probably is not a human being, it will be used by some hardware device or another software component.

(Refer Slide Time: 13:25)



Function points for embedded systems cont...

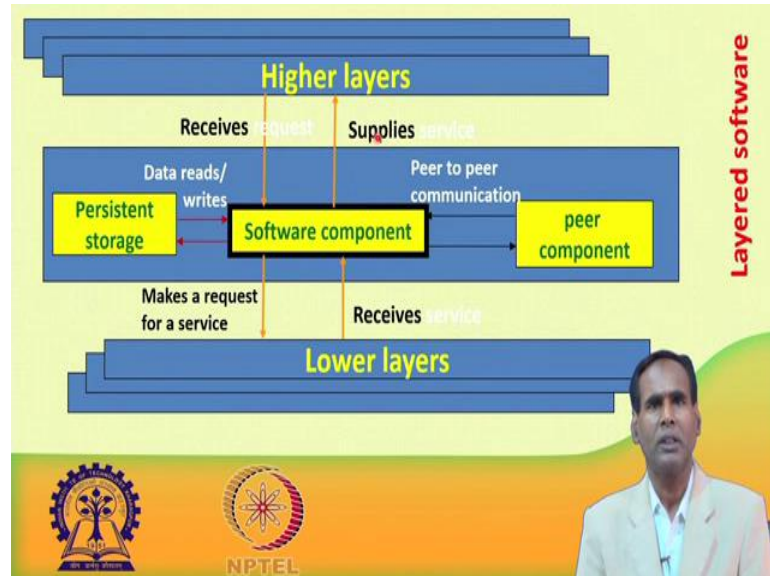
- COSMICS deals with this by decomposing the system architecture into a hierarchy of software layers.
- Embedded software is seen as being in a particular 'layer' in the system
 - Communicates with other layers and also other components at same level
- The software component to be sized, can receive requests for services from layers above and can request services from those below it.

The slide features a yellow background with a green and orange gradient at the bottom. On the left, there are two circular logos: one for IIT Bombay and one for NPTEL. On the right, there is a small inset video of a man in a light-colored suit.

In order to what overcome these problems COSMICS deals with this by decomposing the system architecture into a hierarchy of software layers. And the embedded software is seen as being in one of the or in a particular layer in the system. This layer where this software is present this communicates with the other layers I mean, upper layers and the below layers and also other components are the same level. The software component

which has to be sized, it can receive requests for services from layers above and it can request services from those below it this is shown in the, in this figure.

(Refer Slide Time: 14:05)





So, the software component represents suppose here it may receive what messages from higher layers. It may make a request for a service what from a lower layer and also it may receive a what service from a what a lower layer it may service it may provide some service to some higher layer.

Besides that this software component may communicate with the persistent storage, peer component and other components present in the same layer. This is the layered software and based on this concept this embedded systems work. So, since they Albrecht function point and Symons function points they cannot handle this embedded systems that is why this, because another what function point technology has been developed that is COSMICS Now let us see how COSMICS does work.

(Refer Slide Time: 14:57)

Function points for embedded systems cont...

- This identifies the boundary of the software component to be accessed and thus the points at which it receives inputs and transmits outputs.
- Inputs and outputs are aggregated into data groups, where each group brings together data items that relate to the same object of interest.



So, now, as I have already told you that here the software component that has to be sized can receive requests for services from layers above and it can requests services from those below it. So, this identifies what will the boundary of the software component that has to be assessed and thus the points at which it receives inputs and transmits outputs. It inputs; here, the inputs and outputs are aggregated into data groups where each group will bring together data items that relate to the same object of interest.


(Refer Slide Time: 15:37)

COSMIC FPs

Data groups can move about in 4 ways as follows.

- **Entries:** movement of data into software component from a higher layer or a peer component
- **Exits:** movements of data out
- **Reads:** data movement from persistent storage
- **Writes:** data movement to persistent storage

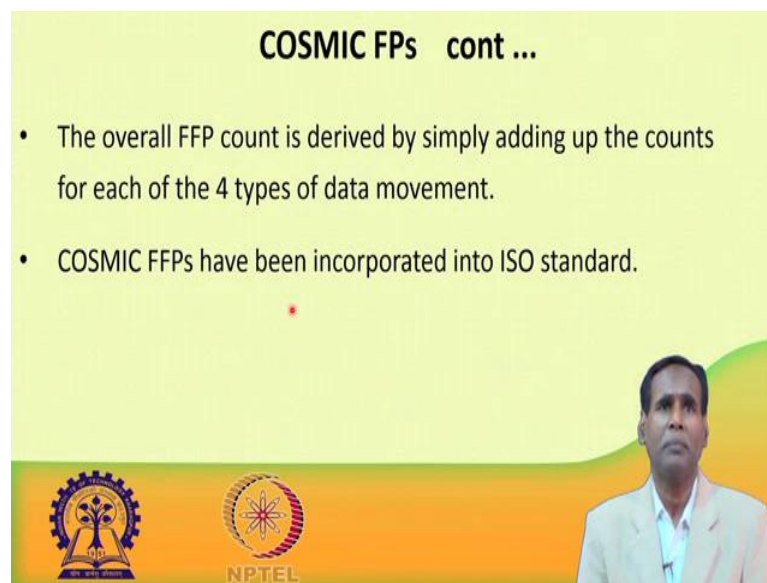
Each counts 1 'COSMIC functional size unit' (Cfsu).



Let us see, what can these; what groups the data groups can move about in four ways as follows and like one is entry another is exits, another is reads, another is writes. So, these are the what, data groups these are the four ways in which the data groups can move about. So, entries means movement of data into the software component that is under consideration from a higher layer or a peer component. Exit means, movement of data out of the software component that is under consideration. Read means, data movement from persistent storage and writes means, data moment to persistent storage.

So, in this way before which the data groups can move and what we have to do that each counts; so, each of the above each counts one COSMIC functional size unit and in short we call as Cfsu. So, in this way the COSMIC function points can be a calculated in this. So, the COSMIC function points can be computed, can be estimated for any embedded system or real time system.

(Refer Slide Time: 16:43)



COSMIC FPs cont ...

- The overall FFP count is derived by simply adding up the counts for each of the 4 types of data movement.
- COSMIC FFPs have been incorporated into ISO standard.

The slide features a yellow background with a green and orange gradient at the bottom. On the left, there are two logos: the Indian Institute of Technology (IIT) logo and the NPTEL logo. On the right, there is a small inset image of a man in a light-colored suit.

The overall full function point count, how it can be found out the overall full function point, count can be derived by simply adding up the counts for each of the 4 types of data moment, that we have seen in the last slide. These COSMIC FFPs they have been incorporated into ISO standard. We have already seen that these LOCs etcetera they are not included incorporated in any ISO standard, but this COSMIC FFPs they have been incorporated into ISO standard.

(Refer Slide Time: 17:16)

COSMIC FPs cont ...

Cons:

- Does not take into account any processing of the data groups once they have been moved into the software component.
- Not recommended for use in systems involving complex mathematical algorithms.

Now, let us quickly say about the, what disadvantages of the cosmic FPs. It does not take into account any processing of the data groups ok. So, it does not take; it does not consider any processing of the data groups once they have been moved into the software component. So, once they what data groups they have been moved into the software component, this metric does not take care of this material, does not take into account any processing of these data groups. Normally it is not recommended for use in systems which involve complex mathematical algorithms. So, this is not recommended for this system. So, these are important drawbacks of cosmic function points.

(Refer Slide Time: 18:00)

Function Points: Pros and Cons

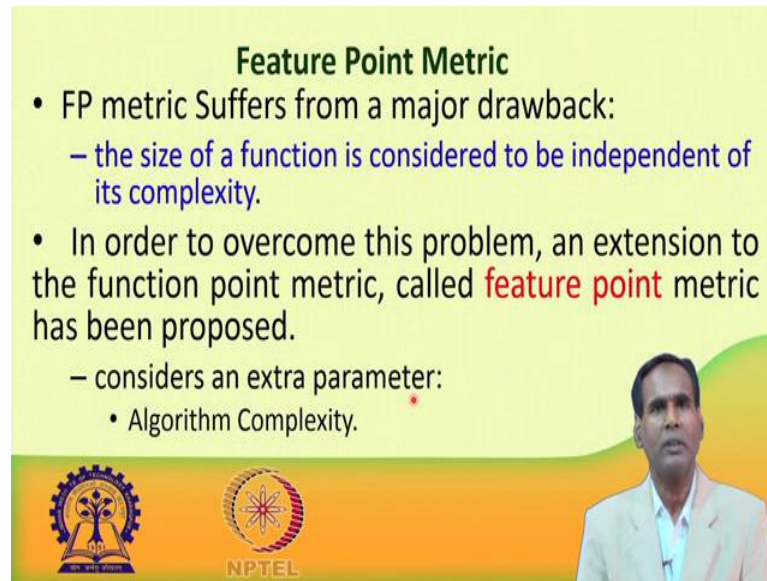
| | |
|---|---|
| <ul style="list-style-type: none">• Pros:<ul style="list-style-type: none">– Language independent– Understandable by client– Simple modeling– Hard to fudge– Visible feature creep | <ul style="list-style-type: none">• Cons:<ul style="list-style-type: none">– Labor intensive– Extensive training– Inexperience may result in inaccuracy– Weighted to file manipulation and transactions– Errors may be introduced by single person, multiple raters advised...– Does not consider algorithmic complexity of a function. |
|---|---|

So, Now let us summarize what are the pros and cons of the function points. On the positive side you can say that function point is language independent, we have already seen LOC is a language dependent, but function points are language independent. Understandable by the client because here no technical details are there, you just what are the functionalities and they are associated things you are just analyzing. So, it can be understandable by clients which are non technical people, it is a very simple modeling technique, hard to fudge and the variable features they creep.

And on the flip side you will see that it is very much labor intensive that to compute the what, function points, extensive training is required, inexperience may result in inaccuracy. So, the; now the function points you have computed if you are quite inexperienced, it may result in inaccurate number of function points, weighted to file manipulation and the transactions, there may be errors which are introduced by single person. So, multiple raters are advised. So, if you adjust giving only one person to estimate this some their sometimes that errors may be introduced. So, you have to put a multiple what raters to rate these what applications, they can find out the different weights so, multiple raters are required.

It does not consider one of the major problem with function point is that it does not consider algorithm complexity of a function. So, because you know that it rates that if there are some functions every function is handled in scenario, but you know some of the functions are more complex, you have to put more effort. some of the functions are very straight forward straight forward functions. So, they require only what less effort or less time. So, it does not consider algorithm complexity of function. So, iterates all the functions similarly.

(Refer Slide Time: 19:53)



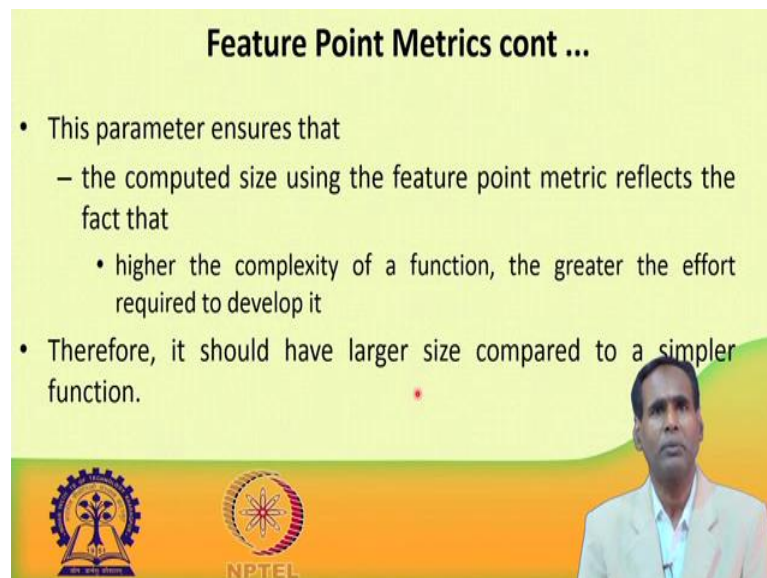
Feature Point Metric

- FP metric Suffers from a major drawback:
 - the size of a function is considered to be independent of its complexity.
- In order to overcome this problem, an extension to the function point metric, called **feature point** metric has been proposed.
 - considers an extra parameter:
 - Algorithm Complexity.

The slide features a speaker overlay on the right side and logos for IIT Bombay and NPTEL at the bottom.

So, I have already told you that function point metric suffers from a major drawback; that means, the size of a function is considered to be independent of its complexity which is not true, the complexity of the functions are different. In order to overcome this problem, an extension to the function point metric is there which is known as feature point metric. So, this feature point metric it takes into account and extra parameter it considers an extra parameter that is known as algorithmic complexity.

(Refer Slide Time: 20:22)



Feature Point Metrics cont ...

- This parameter ensures that
 - the computed size using the feature point metric reflects the fact that
 - higher the complexity of a function, the greater the effort required to develop it
- Therefore, it should have larger size compared to a simpler function.

The slide features a speaker overlay on the right side and logos for IIT Bombay and NPTEL at the bottom.


So, this parameter algorithm complexity it ensures that the computed size using the feature point metric, it reflects the following fact. What if the fact that the higher the complexity of a function the greater the effort required to develop; if they complexity of function is high so, you have to put more effort to develop it. If the complexity of a function is very less just like a small GUI kind of a function it remain require less effort. Therefore, it should have larger size compared to a simpler function.

So, if a; what the function is having higher complexity it should have larger size as compared to a simpler function. It should have larger function point and hence, it should have larger size as compared to a simpler function.

(Refer Slide Time: 21:08)

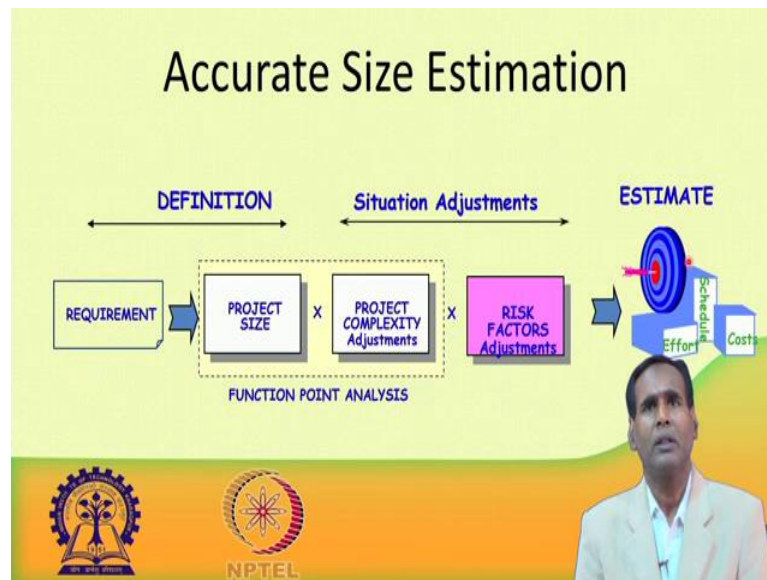
FP/SLOC Conversion

| Language | Median SLOC/function point |
|--------------|----------------------------|
| C | 104 |
| C++ | 53 |
| HTML | 42 |
| JAVA | 59 |
| Perl | 60 |
| J2EE | 50 |
| Visual Basic | 42 |



So, finally, let us see what the relationship or how that is the relationship between function points or SLOC or how a given a function point, how it can be converted to SLOC? See, for different programming languages this conversion rate is different like for C language 104 SLOC is there per 1 function point. So, 1 function point may contain 104 SLOC. Similarly for C plus plus the SLOC for function point is 53 and like this, you can see it is highest in case of C and we can see that it is almost lowest in case of HTML and visual basic it is 42 ok.

(Refer Slide Time: 21:54)



So, how you can still make your estimation more accurate, how can do accurate size estimation. So, here **1** one thing you have to extra take that is the risk factors, because in the previous things almost we have neglected the risks factors. So, in order to make your size estimation more accurate, you have to take the risk factors and then you have to multiply.


So, you will take the requirement then estimate the project size, then multiplied with the project complexity adjustment factors, then multiplied with the risk factors. So, these two points the we have already told you this is given where in the function point analysis, the project size and the project complexity adjustment. This you can say the unadjusted function points and these are the complexity factors by multiplying this you can get some value and that will multiplied by the risk factor.

So, these two factors can be obtained from function point analysis, then by what under these two factors from the definition and the project completion adjustment and risk factors they are dependent. The situation you have to adjust the situation finally, you can if you will use all those things then by taking the multiplication you can estimate what are the cost the effort and you can perform the scheduling.

(Refer Slide Time: 23:14)

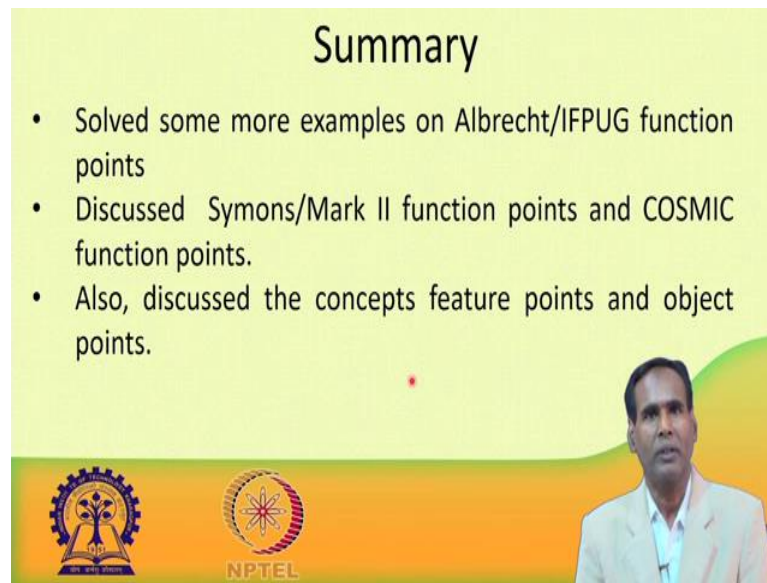
Object Points

- Object points has nothing to do with object-oriented programming
- Number of object points is estimated based on
 - Number of separate screens displayed
 - Number of reports that are produced
 - Number of modules in the code •
- Object points are simpler to estimate & GUI into account



Finally, let us quickly say about the object points. See do not confuse objects points, they have nothing to do with the object oriented programming. Here only the number of object points is estimated; the number of object points is estimated based on three factors. First the number of separate screens they are displayed, then the number of reports that are produced and the number of modules they are present in the code. By taking into account, then by using some formula you can find out the object points. These object points are normally are the; they are usually simpler to estimate and they take into account the graphical user interfaces. So, this is a little bit fundamental about object points.



(Refer Slide Time: 23:57)




Summary

- Solved some more examples on Albrecht/IFPUG function points
- Discussed Symons/Mark II function points and COSMIC function points.
- Also, discussed the concepts feature points and object points.

•



So, finally, we have seen that we have solved two more problems on Albrecht IFPUG function points, then we have discussed the Symons Mark II function points and the cosmic function points. We have seen that how cosmic point it is already included in this, what ISO standards also we have discussed what the concept of feature point because we have seen that function point has one important a drawback. That it does not take into account to the algorithmic complexity if two different functions are there.

It treats them as equally, but in fact, the two function those are there one might be very highly complex, another might be very less complex. So, which one is highly complex we have to put more effort and which one is less complex we have to put less effort. So, these what complexity is not taken into account by function points; so, that is why function point has been extended to in consider this complexity, it has another parameter called as algorithmic complexity.

So, this algorithm complexity takes into account that if the complexity of a function is more, more effort should be given if the complexity of a function is less. So, less effort has to be given. So, if the complexity of a function is more; obviously, it should contain it should produce or it should have more number of function points and hence, the size will be more. And if the function is having less complexity, then it should have less function points and hence less size. Also we have discussed little bit a of the object

points which takes an account or three factors that or three parameters ok, object points it takes into three; it is based on three parameters that also we have discussed.

So, these are the things that we have discussed on today. So, in the next class we will see and that COCOMO model.

(Refer Slide Time: 26:03)



We have taken from these books, the reference mainly what it is both of the books we have used for this topics and finally, we stop here.

Thank you very much.