Cost Estimation: When we talk about cost, we need to consider the time experts need to spend on the project to finish it. Knowing this will allow us to know how much we will be paying them.

Effort Estimation: We need to calculate the effort in person-months for a project. The salary is assumed to be paid per month or per hour, depending on how many hours the expert can work and the organization's policies.

Additional Costs: In general, we add a certain percentage (e.g., 20%) to the person-months times the monthly salary. This amount is reserved for other costs, which can sometimes be more. These costs include the budget for the match, travel, hardware, software, tools, licenses, etc.

Users are Expensive: Users are often the most expensive because you need to pay experts and developers.

Project Initiation Phase: We talked about the initiation phase of any project, identified what a charter is, what a scope is, how we define objectives, and what project activities are performed during the project.

Effort Estimate Techniques: In Chapter 3, we will be talking about what an effort estimate is, the different techniques that we can follow, how a cost estimate for a project is made, the different cost estimation techniques, how a schedule estimate for a project is made, and how a resource estimate is made.

Estimation Uncertainty: It's good to have a range of time saying that it takes one year, two years, one month, one week, two weeks, three days. Starting with nothing at all does not happen. However, because it is an estimate we always start with a certain value, but it is always not the real value. We are at least 25 percent, if not more, plus or minus as far from there.

Effort Estimation Method: An effort estimate for a project is made in general by collecting data from experience.

Estimation Techniques: There are different techniques for effort estimation. Some of them include algorithmic modeling (like COCOMO), experience-based techniques (asking an expert), and group consensus techniques (like Delphi).

Algorithmic Modeling: This technique uses a formula that considers various factors such as the development team's effort, velocity, productivity, the context in which the application is developed, the domain of the application, its complexity, and the number of functionalities.

Experience-Based Techniques: These techniques rely on experts who provide their estimates based on their experience. They might consider factors like how much time it takes to create or access a database, or the proportion between the time it takes to develop a certain number of lines of code.

Group Consensus Techniques: In these techniques, each expert alone gives an estimate. After this, they sit together to discuss their estimates, review them, and come up with numbers.

Delta Technique: This technique considers the fastest time the development can be done (optimistic), the longest time it could take (pessimistic), and the average time. A formula is used where one for the pessimistic and one for the optimistic and four for the average are considered. After this, we divide by six.

Role of Project Manager: As a project manager, it's your duty to estimate the time, average, best, and worst-case scenarios. However, sometimes there are some companies who estimate right.

Role of Project Manager: The project manager is not supposed to be the one person doing and in charge of everything. The project manager should know some of the work and rely on experts for the rest, especially when it comes to estimation.

Estimation Techniques: If the manager considers themselves an expert, they can choose to follow any of the estimation techniques like the Delphi technique or the function points application. The Delphi technique involves inviting persons to provide their estimates, while the function points application involves applying a formula.

Outsourcing Estimation: A project manager can choose to outsource the cost estimation to a company. However, it's important to trust the company and know what you want.

Data Collection: The major challenge is not which technique to follow, but how to collect the data to be able to apply the technique.

Function Points Application: This technique relies on giving some project data. It's a formula that you apply.

Learning Objective: The learning objective of the course is to know what exists and how to apply it. It's important to remember that we have done for, what is that?

Effort Estimation: When you finish the course, if you are asked what is effort estimation, you should be able to understand what it is, and what the techniques are, such as Delphi, function points, and analysis by analogy.

Function Points Technique: This technique involves identifying the function points of the current project. This value will be used to generate a certain number of lines of code. For this number of lines of code, we will be telling how many person-months are needed.

Analyzing the Current Project: To generate the line of code, we need to have previous data with some function point values that were associated with a certain number of lines of code. We also need information about the current project.

Counting Inputs, Outputs, Files, Interfaces, and Inquiries: When we take an application, we need to have a certain count of files, input files and output files, input data and output data, internal files and external files, interfaces, and inquiries. For each of these counts, we need to have a complexity weight.

Unadjusted Function Point Count: We generate what we call the unadjusted function point count. It's only a simple count based on these data and put output interfaces.

Complexity Multiplier: Once all this number is generated, we need to put this application or count in its context of complexity. We did not consider any security issue, availability, online interaction issue, or any of the non-functional characteristics of the architecture. For these, we also generate a complexity multiplier.

Function Points: We multiply this unadjusted function points according to a certain formula and we generate the function points. For the function points, we can calculate how many lines of code are needed.

Estimation Based on Application: When generating an estimate, you need to consider the specific application you are developing. This could be a new application, a prototype, an application where you already have the design or analysis, or an application where you are reusing some existing parts. Depending on the scenario, you will come up with an effort estimate.

COCOMO: In the COCOMO (Constructive Cost Model), you can tune the model based on the application you are developing. The effort estimate will differ depending on whether it is a development from scratch or reusing some existing components.

Software for Estimation: There are different software tools available for each of the estimation techniques. For example, there is software for the Delphi technique, function points application, and others.

Importance of Effort Estimate: An effort estimate is very important as it is an indicator of success. It helps in managing the budget and ensuring customer satisfaction.

Challenges in Estimation: Estimating at the beginning is not easy. You need to have a lot of experience, but even then, you are far from the current event. It comes with experience, but you cannot say that you are 100% sure.

Experience-Based Technique: Sometimes managers will not bother doing the details of how the budget or the effort needed is estimated because they assume that they know the team and the work. This works mainly when it is an exact project.

Domain-Specific Companies: Most of the companies, especially development companies, are engaged within a specific field. So they work on a specific kind of development or another kind of project. Their businesses are domain-specific. So, over time, they gain more experience.

Estimation Based on Context: In most cases, you need to understand the context of the project and the customer's needs. The estimation process begins even before the initiation phase, and as we progress, the estimates become more accurate.

Estimation Techniques: Many techniques exist for effort estimation, such as function point analysis, Delphi, and COCOMO. The choice of technique depends on the availability of data, the calendar, and the previous step.

Choosing Suitable Technique: A suitable effort estimate technique is chosen for any specific project. Since effort estimation techniques are not perfect, the effort estimate needs to be revised as the project progresses.

Expert Input: Experts are needed to have accurate cost estimates and effort estimates. It's good to take note or record your experience with the effort estimate people, because for future projects, you will be using the data from this project.

Analogy Technique: One technique that doesn't rely on an algorithm but follows a certain logic is the analogy technique. You can estimate a new project by comparing it to similar past projects. This involves dissecting the current application and defining the different effort and different activities, and saying that this is similar to that.

Determining Size Using Analogy: Using analogy to determine size involves starting with a certain project and then identifying the specifications and requirements. Then, you compare this to another project, identifying similarities.

Detailed Size Results: The analogy technique generates detailed size results for similar previous projects. You can look into subsystems and activities related to various aspects of the project. For example, you can compare how many databases you have in the previous project and the current project.

Estimation Based on Application: When generating an estimate, you need to consider the specific application you are developing. This could be a new application, a prototype, an application where you already have the design or analysis, or an application where you are reusing some existing parts. Depending on the scenario, you will come up with an effort estimate.

COCOMO: In the COCOMO (Constructive Cost Model), you can tune the model based on the application you are developing. The effort estimate will differ depending on whether it is a development from scratch or reusing some existing components.

Software for Estimation: There are different software tools available for each of the estimation techniques. For example, there is software for the Delphi technique, function points application, and others.

Importance of Effort Estimate: An effort estimate is very important as it is an indicator of success. It helps in managing the budget and ensuring customer satisfaction.

Challenges in Estimation: Estimating at the beginning is not easy. You need to have a lot of experience, but even then, you are far from the current event. It comes with experience, but you cannot say that you are 100% sure.

Experience-Based Technique: Sometimes managers will not bother doing the details of how the budget or the effort needed is estimated because they assume that they know the team and the work. This works mainly when it is an exact project.

Domain-Specific Companies: Most of the companies, especially development companies, are engaged within a specific field. So they work on a specific kind of development or another kind of project. Their businesses are domain-specific. So, over time, they gain more experience.

Function Points Technique: This technique involves identifying the function points of the current project. This involves analyzing the functionalities of the project and the complexity of these functionalities.

Counting Elements: The functionalities are categorized into five elements: input, output, files, interfaces, and inquiries. Each of these elements is counted based on the data and output interfaces of the project.

Assigning Complexity Weights: Each of these counts is assigned a complexity weight. The weights are determined based on factors such as the type of input (e.g., integer, barcode, file) and the complexity of handling that input.

Generating Unadjusted Function Point Count: The complexity weights are used to generate what is called the unadjusted function point count. This is a simple count based on the data and output interfaces.

Considering Non-Functional Characteristics: The application or count is then put in its context of complexity, considering factors such as security, availability, online interaction, and other non-functional characteristics of the architecture.

Generating Complexity Multiplier: A complexity multiplier is generated based on answers to 14 questions about factors like security, online availability, internet dependency, etc. Each question is answered on a scale of zero to five.

Calculating Function Points: The unadjusted function points are multiplied by the complexity multiplier according to a certain formula to generate the function points.

Estimating Lines of Code: The function points can then be used to estimate how many lines of code are needed for the project. This requires previous data with function point values associated with a certain number of lines of code, as well as information about the current project.

Estimation Based on Development Process: The estimation process can vary depending on the development process being followed. For instance, if you're following the waterfall model, you might estimate differently than if you're following an iterative model.

Skills and Experience: When estimating, it's important to consider the skills needed for the project and the experience of the team. This includes both technical skills and soft skills like communication and scheduling.

Function Points: Function points are a unit measure for software much like hours are to measuring time, miles are to measuring distance or Celsius is to measuring temperature. In the function points technique, you need to consider different scenarios and know what the development lifecycle is to generate an accurate estimate.

Using Reusable Components: If a project is using a number of reusable components from previous projects, this should be considered in the estimate as it can reduce the cost.

Delphi Technique: The Delphi technique is a method for structuring a group communication process so that the process is effective in allowing a group of individuals, as a whole, to deal with a complex problem. In the Delphi technique, you invite experts to give their estimates, then everyone sits together to discuss and come to a consensus.

Estimation is Complex: Estimation is a complex task that involves considering many different factors, including the size and complexity of the project, the skills and experience of the team, and the development process being used. It's not just about counting the number of lines of code or function points.

Function Points Technique: This technique involves identifying the function points of the current project. This involves analyzing the functionalities of the project and the complexity of these functionalities.

Counting Elements: The functionalities are categorized into five elements: input, output, files, interfaces, and inquiries. Each of these elements is counted based on the data and output interfaces of the project.

Assigning Complexity Weights: Each of these counts is assigned a complexity weight. The weights are determined based on factors such as the type of input (e.g., integer, barcode, file) and the complexity of handling that input.

Generating Unadjusted Function Point Count: The complexity weights are used to generate what is called the unadjusted function point count. This is a simple count based on the data and output interfaces.

Considering Non-Functional Characteristics: The application or count is then put in its context of complexity, considering factors such as security, availability, online interaction, and other non-functional characteristics of the architecture.

Generating Complexity Multiplier: A complexity multiplier is generated based on answers to 14 questions about factors like security, online availability, internet dependency, etc. Each question is answered on a scale of zero to five.

Calculating Function Points: The unadjusted function points are multiplied by the complexity multiplier according to a certain formula to generate the function points.

Estimating Lines of Code: The function points can then be used to estimate how many lines of code are needed for the project. This requires previous data with function point values associated with a certain number of lines of code, as well as information about the current project.

Identifying Function Points: The first step in the Function Points Application technique is to identify the function points of the current project. This involves analyzing the functionalities of the project and the complexity of these functionalities.

Counting Elements: The functionalities are categorized into five elements: input, output, files, interfaces, and inquiries. Each of these elements is counted based on the data and output interfaces of the project.

Assigning Complexity Weights: Each of these counts is assigned a complexity weight. The weights are determined based on factors such as the type of input (e.g., integer, barcode, file) and the complexity of handling that input.

Generating Unadjusted Function Point Count: The complexity weights are used to generate what is called the unadjusted function point count. This is a simple count based on the data and output interfaces.

Considering Non-Functional Characteristics: The application or count is then put in its context of complexity, considering factors such as security, availability, online interaction, and other non-functional characteristics of the architecture.

Generating Complexity Multiplier: A complexity multiplier is generated based on answers to 14 questions about factors like security, online availability, internet dependency, etc. Each question is answered on a scale of zero to five.

Calculating Function Points: The unadjusted function points are multiplied by the complexity multiplier according to a certain formula to generate the function points.

Estimating Lines of Code: The function points can then be used to estimate how many lines of code are needed for the project. This requires previous data with function point values associated with a certain number of lines of code, as well as information about the current project.

Considering Non-Functional Characteristics: The application or count is then put in its context of complexity, considering factors such as security, availability, online interaction, and other non-functional characteristics of the architecture.

Generating Complexity Multiplier: A complexity multiplier is generated based on answers to 14 questions about factors like security, online availability, internet dependency, etc. Each question is answered on a scale of zero to five.

Calculating Function Points: The unadjusted function points are multiplied by the complexity multiplier according to a certain formula to generate the function points.

Estimating Lines of Code: The function points can then be used to estimate how many lines of code are needed for the project. This requires previous data with function point values associated with a certain number of lines of code, as well as information about the current project.

COCOMO (Constructive Cost Model): This is a software estimation model that uses a basic formula to estimate project cost, and can be adapted and adjusted to fit individual project needs. The formula is typically in the form of Effort = A * (Size)^B * M, where:
A is a constant factor that depends on the organization and the nature of the project.
Size is the size of the software project, typically measured in lines of code or function points.
B is an exponent that accounts for the diseconomies of scale. As the size of the project grows, there is typically more communication required, more documentation needed, and more coordination needed, which all increase the effort nonlinearly.
M is a multiplier that accounts for the influence of cost drivers on the project.