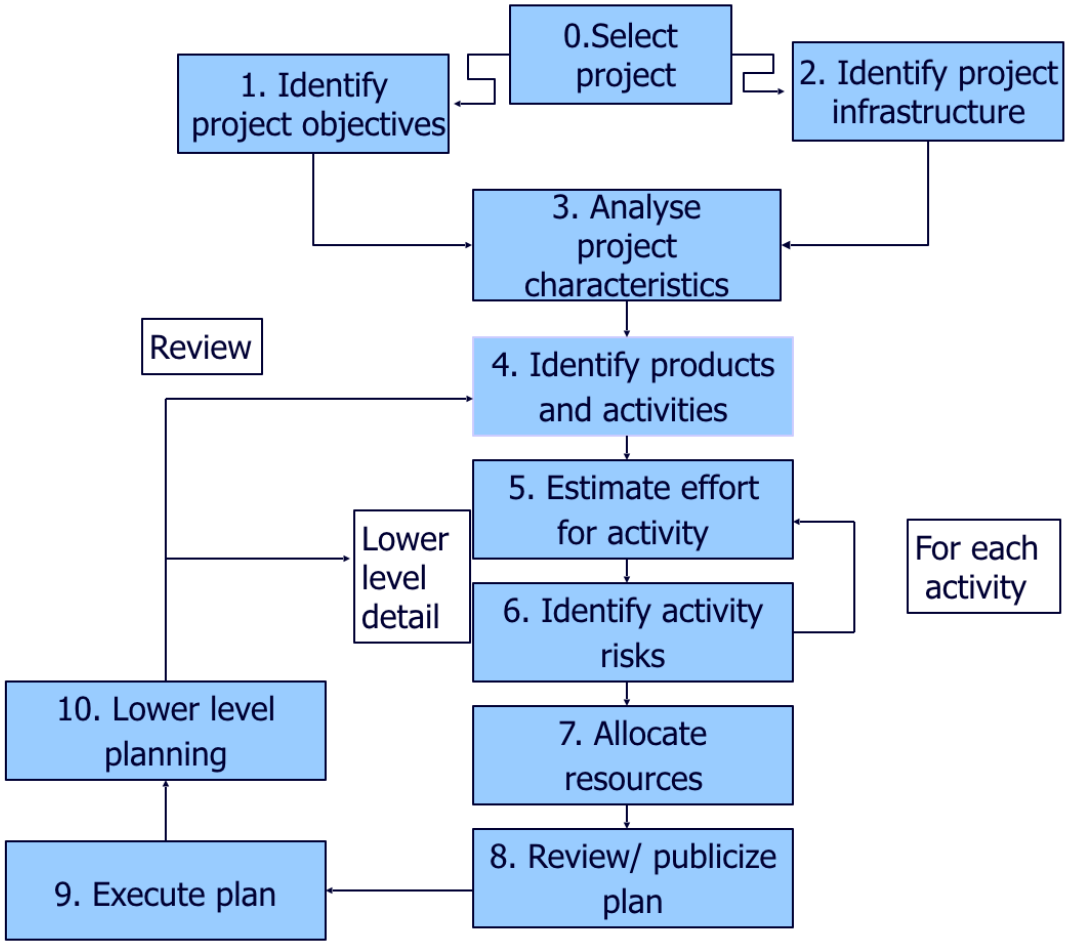
Unit -3 Software Estimation Techniques

Project evaluation is normally carried out in Step 0 stepwise in the following figure. The subsequent steps of step wise are then concerned with managing the development project that stems from this project selection.

Different steps of project:



Software Effort Estimation

Effort estimation is a vital aspect of project management, playing a significant role in setting realistic timelines and allocating resources efficiently. It involves predicting the amount of time and effort required to complete a particular task or project.

Effort estimation is one of the initial steps in software development projects. Although its complexity, when performed right, effort estimation creates a basis for all subsequent stages related to project planning and management.

Choosing technologies

An outcome of project analysis will be the selection of the most appropriate methodologies and technologies. Methodologies include techniques like the various flavors of object-oriented (OO) development, SSADM and JSP (Jackson Structured Programming) while technologies might include an appropriate application-building environment, or the use of knowledge-based system tools.

As well as the projects and activities, the chosen technology will influence the following aspects of a project:

- The training requirements for development staff.
- The types of staff to be recruited.
- The development environment – both hardware and software.
- System maintenance arrangements.

Following are some of the steps of project analysis:

1. **Identify project as either objectives-driven or product-driven:** There will be cases where things are so vague that even the objectives of the project are uncertain or are the subject of disagreement. People may be experiencing a lot of problems, but no one knows exactly what the solution to be problems might be. It could be that the IT specialists can provide help in some places but assistance from other specialists is needed in others. In these kinds of situations, a soft systems approach might need to be considered.
2. **Analyze other project characteristics:** The sorts of question that would need to be asked include the following:
   a. **Is a data oriented or a control-oriented system to be implemented?** 'Data oriented' systems generally mean information systems that will have a considerable database. 'Control oriented' systems refer to embedded control systems. These days it is not uncommon to have systems with components of both types.
   b. **Will the software that is to be produced be a general package or application specific?** An example of a general package would be a spreadsheet or a word processing package. An application specific package could be, for example, an airline seat reservation system.
   c. **Is the system to be implemented of a particular type for which specific tools have been developed?** For example: (i) does it involve concurrent processing? (ii) will the system to be created be knowledge based? (iii) Will the system to be produced make heavy use of computer graphics?

d. **Is the system to be created safety – critical?** For instance, could a malfunction in the system endanger human life?
e. **What is the nature of the hardware/software environment in which the system will operate?** It might be that the environment in which the final software will operate is different from that in which it will be developed. Embedded software may be developed on a large development machine that has lots of supporting software tools in the way of compilers, debuggers, static analyzers and so on, but might then be downloaded to a small processor in the larger engineered product. A system destined for a personal computer will need a different approach to one destined for a main frame or a client server environment.

3. **Identify high level project risks:** When we first embark on a project, we might be expected to work out elaborate plans even though we are at least partially ignorant of many important factors that will affect the project. The greater the uncertainties at the beginning of the project, the greater the risk that the project will be unsuccessful. Once we recognize a particular area of uncertainty we can, however, take steps to reduce its uncertainty. One suggestion is that uncertainty can be associated with the products, processes, or resources associated with the project.
   a. **Project uncertainty:** Here we ask how well the requirements are understood. It might be that the users themselves are uncertain about what a proposed information system is to do.
   b. **Process uncertainty:** It might be that the project under consideration is the first where an organization has tried to use a method. Perhaps a new application building tool is being used. Any change in the way that the systems are developed is going to introduce uncertainty.
   c. **Resource uncertainty:** The main area of uncertainty here will almost surely be the availability of staff of the right ability and experience. A major influence on the degree of uncertainty in a project will be the sheer size of a project. The larger the number of resources needed or the longer the duration of the project, the more inherently risky it is likely to be.
4. **Consider user requirements concerning implementation:** A user organization lays down standards that have to be adopted by any contractor providing software for them.
5. **Select general life cycle approach:**
   a. **Control systems:** A real-time system will have to be implemented using an appropriate methodology.

b. **Information systems:** Similarly, an information system will need a methodology.
c. **General applications:** Where the software to be produced is for the general market rather than for a specific application and user, then a methodology would have to be thought about very carefully.
d. **Specialized techniques:** These have been invented to expedite the development of, for example knowledge-based systems where there are a number of specialized tools and logic-based programming languages that can be used to implement this type of system.
e. **Hardware environment:** The environment in which the system is to operate can put constraints on the way it is to be implemented.
f. **Safety-critical systems:** Where safety and reliability are of the essence, it might be possible to justify the additional expense of a formal specification using a notation.
g. **Imprecise requirements:** Uncertainties or a novel hardware/software platform may mean that a prototyping approach should be considered. If the environment in which the system is to be implemented is a rapidly changing one, then serious consideration would need to be given to incremental delivery.

Choice of process model

The word 'process' is sometimes used to emphasize the idea of a system in action. To achieve an outcome, the system will have to execute one or more activities: this is its process. The idea can be applied to the development of computer-based systems where several interrelated activities have to be undertaken to create a final product. These activities can be organized in different ways and we can call these process models.

Effort Estimation Techniques in Software Testing:

In the range of software testing, effort estimation plays a crucial role in planning and executing testing activities. Determine the resources, time, and budget needed to successfully execute a software testing project with the aid of effort estimating techniques.

Testing teams may optimize their workflow, allocate resources wisely, and produce high-quality software within the set deadlines by precisely calculating the effort required.

1. **Expert Judgment**
   Expert judgment, which draws on the skills and expertise of seasoned experts in the field of software testing, is a frequently used technique in work estimation. Testing teams can learn a lot about the time needed for different testing activities by talking to professionals who have worked on comparable projects in the past. To give accurate estimates, these specialists do thorough analyses of variables such as project complexity, scope, and testing team members' skills. Expert judgment improves the precision of effort estimation by minimizing uncertainty.
2. **Analogous Estimation**
   Analogous estimate, commonly referred to as historical or top-down estimating, compares the ongoing software testing effort to earlier initiatives that have already been completed. This method makes the assumption that similar projects will take a similar amount of effort. Testing teams can estimate the effort for the present project by looking at historical data and benchmarking against earlier initiatives. When there is a lack of specific project details or a shortage of estimation time, analogous estimation is particularly helpful.
3. **Parametric Estimation**
   A quantitative method called parametric estimation uses statistical analysis and mathematical models to calculate effort. Through this method, a connection is made between project metrics including software size, project complexity, and testing team productivity rate and effort. By taking into account these variables, parametric estimation determines effort using pre-established formulas or algorithms. For this method to produce accurate estimates, data must be collected and analyzed precisely.
4. **Three-Point Estimation**
   The three-point estimate method approaches effort estimation more probabilistically. It entails taking into account three effort estimates: the most likely estimate (M), the pessimistic estimate (P), and the optimistic estimate

(O). Using the formula (O + 4M + P) / 6, the anticipated effort is then determined using these estimates. The three-point estimating technique offers a more thorough and risk-aware effort assessment by taking into account the best-case, most likely, and worst-case scenarios.

5. **Function Point Analysis**
   A common method for estimating work is called Function Point Analysis (FPA), especially in projects that use a structured development approach. FPA assesses the functionality offered by the software and weights various components according to their complexity. FPA assists in estimating the time needed to design, implement, and test the software by quantifying the functionality. This method necessitates a deep comprehension of the program specifications as well as the capacity to disassemble a system into its working parts.
6. **Bottom-Up Estimation**
   The testing work is divided into smaller pieces, such as test cases or modules, and each unit's testing effort is estimated. The total effort for the testing activity is then calculated from the individual estimations that have been pooled. When there is a clear understanding of the testing components and their effort needs, the bottom-up estimate is helpful.
7. **Top-Down Estimation**
   Top-Down estimation starts with an overall estimate for the testing activity and then allocates effort to different components or phases based on expert judgment or historical data. It is a high-level estimation approach that provides an initial estimate for planning purposes. Top-Down estimation is useful when there is limited detailed information available or when time constraints are tight.
8. **Wideband Delphi Technique**
   The Wideband Delphi technique combines the benefits of the Delphi technique with group decision-making. In this technique, a group of experts provides individual estimates anonymously. Following the sharing of the estimates, experts participate in a facilitated discussion to discuss the estimates. Up until an agreement is obtained, the process is iterative. The Wideband Delphi approach improves communication and knowledge sharing while minimizing estimating bias.