**Course Name: Software Engineering Economics**

**Credit Hours:** 3

**Prerequisites: (Intro. to) Software Engineering**

**Objectives:** Determine how new software development technologies affect the economics and risks of software development. Understand and characterize how the paradigm shift affects or replaces our current methods of software cost, schedule and risk estimation. Identify best practices and lessons learned with Web-based developments. Identify acquisition and lifecycle risks

**Outcomes:** The student will be able to learn,

1. Understand and be able to apply the key software engineering economic fundamentals to real-world software economic issues. 2. Illustrate through example the key software life cycle economics, including product and process life cycles; portfolios; proposals; investment decisions; pricing and costing, and earned value management (EVM). 3. Apply the concepts of risk and uncertainty to real world software development projects, including goals; estimates; prioritization and decision making. 4. Perform best-practice economic analysis methods. 5. Relate and interpret the “good-enough” principle; friction-free economy; ecosystems and outsourcing

**Course Outline:**

Programming aspects, economic aspects, human relations aspects, software trends: cost, social impact, the plurality of SE Means, The GOALS Approach to Software Engineering, The Software Work Breakdown Structure (WBS), Software Maintenance, introduction to COCOMO, definitions and assumptions, development effort and schedule, phase distribution, The Raylaigh Distribution, interpolation, basic software maintenance effort estimation. Performance Models, Optimal Performance, Sensitivity Analysis, Cost-Effectiveness Models.

**Reference Materials:**

1. *Software Engineering Economics*, Boehm, Prentice Hall, 1981(or Latest Edition).

2. *Software Cost Estimation with COCOMO II*, Boehm et al., Prentice Hall, 2000 (or Latest Edition).

3. *Making the Software Business Case: Improvement, Reifer, Don*, Addison Wesley, 2001, (or Latest Edition).

**Course Outline:**

Economic aspects of Software Engineering, programming aspects of Software Engineering, human relations aspects of Software Engineering, COCOMO,

General views of software engineering, software development effort,Software failure, software maintenance estimation, software development phase distribution, software trends: cost, social impact, Cost-Effectiveness Model.

***1). Economic aspects of Software Engineering:***

***Economics*** *is the study of how resources (people, time, facilities, money) are used to produce and distribute commodities and how services are provided in society.*[*Engineering economics*](https://en.wikipedia.org/wiki/Engineering_economics)*is a branch of microeconomics dealing with engineering related economic decisions.*

Software engineering economics is about making decisions related to software engineering in a business context. The success of a software product, service, and solution depends on good business management. Yet, in many companies and organizations, software business relationships to software development and engineering remain vague. This knowledge area (KA) provides an overview on software engineering economics. Economics is the study of value, costs, resources, and their relationship in a given context or situation. In the discipline of software engineering, activities have costs, but the resulting software itself has economic attributes as well. Software engineering economics provides a way to study the attributes of software and software processes in a systematic way that relates them to economic measures.

**Software Engineering Economics Fundamentals**

### ***Finance***

### ***Accounting***

### ***Controlling***

### ***Cash Flow***

### ***Decision-Making Process***

### ***Valuation***

***2). software development effort:***

The most important activity in software project management process is the estimation of Software development effort. The literature shows many algorithmic cost estimation models such as Boehm’s COCOMO, Albrecht's Function Point Analysis, Soft computing based techniques etc. but each model have their own advantages and disadvantages in predicting development cost and effort. This is because of the availability of project data in the initial stages of development process is often incomplete, inconsistent and vague.

# 5 Steps to Software Development Effort Estimation

## 1- Scoping

You need first to scope the project even if you do not have the full detailed requirements but you can assume some of them or add margins later. While in most cases you will have a defined scope to start with.

You can always list your assumptions to justify the outcome of the estimation process and its results.

## 2- Decomposition

In this step, you will need to break your software into smaller components and functions and you can categorize them to a different set of elements, this is similar to work breakdown structure but only for the software components not all the working activities for the software.

You may also collect different data from the project team or the customer to ensure that you have listed all functionalities.

## 3- Sizing

In this step, the actual estimation will be done for each component alone, and I will illustrate more about how you will do that using the techniques mentioned above, this will be illustrated in 8 steps in details below.

In this step, and for more validation, you can use different estimation techniques to analyze the different estimation outputs and you may take an average of these estimates as well.

## 4- Expert and Peer Review

After initial estimate, you will need at some point to ask for expert opinion for some new functionalities you may not aware off, or for considering a review from your peers that you have done the correct estimation. Moreover, you may need to do some analogy based techniques for similar components or functions developed before or maybe a similar project to ensure that you are on the correct path.

## 5- Estimation Finalization

This can be considered the final step as you aggregate all the estimations from all components and functions and have a baseline estimate. You can go another round across the process until reaching the correct estimate which will be approved by the Project team and the Management as well.

***3) Software failure***

Most software projects fail completely or partial because they don’t meet all their requirements. These requirements can be the cost, schedule, quality, or requirements objectives. According to many studies, failure rate of software projects ranges between 50% – 80%. There are a variety of causes for software failures but the most common are:

* Lack of user participation
* Changing requirements
* Unrealistic or unarticulated project goals
* Inaccurate estimates of needed resources
* Badly defined system requirements
* Lack of resources
* Poor communication among customers, developers, and users
* Use of immature technology
* Inability to handle the project’s complexity
* Poor Project Management
* Lack of Stakeholder involvement

***4) Software maintenance estimation***

Reports suggest that the cost of maintenance is high. A study on estimating software maintenance found that the cost of maintenance is as high as 67% of the cost of entire software process cycle.

On an average, the cost of software maintenance is more than 50% of all SDLC phases. There are various factors, which trigger maintenance cost go high, such as:



### **Real-world factors affecting Maintenance Cost**

* The standard age of any software is considered up to 10 to 15 years.
* Older softwares, which were meant to work on slow machines with less memory and storage capacity cannot keep themselves challenging against newly coming enhanced softwares on modern hardware.
* As technology advances, it becomes costly to maintain old software.
* Most maintenance engineers are newbie and use trial and error method to rectify problem.
* Often, changes made can easily hurt the original structure of the software, making it hard for any subsequent changes.
* Changes are often left undocumented which may cause more conflicts in future.

### **Software-end factors affecting Maintenance Cost**

* Structure of Software Program
* Programming Language
* Dependence on external environment
* Staff reliability and availability

## Maintenance Activities

IEEE provides a framework for sequential maintenance process activities. It can be used in iterative manner and can be extended so that customized items and processes can be included.

