



# Part 4

# Managing Software Project

## (软件项目管理)

**Software Engineering: A Practitioner's Approach, 6th edition**  
*by Roger S. Pressman*



# 概述

- 这部分将考虑计划、组织和监控软件项目所需的管理技术：
  - 人员、过程和问题管理
  - 软件项目和软件过程管理
  - 软件项目的工作量、成本和工期估算
  - 风险评估
  - 选择软件工作任务集
  - 进度计划
  - 质量管理
  - 正式技术评审
  - 变更管理

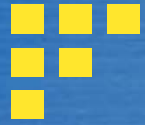


# Chapter 21

## Project Management

**Software Engineering: A Practitioner's Approach, 6th edition**  
*by Roger S. Pressman*





# 本章要点

---

- 理解4P



# The 4 P's

- **People** — the most important element of a successful project
- **Product** — the software to be built
- **Process** — the set of framework activities and software engineering tasks to get the job done
- **Project** — all work required to make the product a reality



# 1 People

---

- Stakeholders
- Team Leader
- Software Team
- Team Coordination & Communication





# Stakeholders

- **Senior managers** who define the business issues that often have significant influence on the project.
- **Project (technical) managers** who must plan, motivate, organize, and control the practitioners who do software work.
- **Practitioners** who deliver the technical skills that are necessary to engineer a product or application.
- **Customers** who specify the requirements for the software to be engineered and other stakeholders who have a peripheral[外围的] interest in the outcome.
- **End-users** who interact with the software once it is released for production use.



# Software Teams

How to lead?

How to organize?

How to collaborate?



How to motivate?

How to create good ideas?





# Team Leader

- The **MOI** Model:
  - **Motivation[激励]**. The ability to encourage (by “push or pull”) technical people to produce to their best ability.
  - **Organization[组织]**. The ability to mold[塑造] existing processes (or invent new ones) that will enable the initial concept to be translated into a final product.
  - **Ideas or innovation[思想或创新]**. The ability to encourage people to create and feel creative even when they must work within bounds established for a particular software product or application.



# Software Teams

## Selecting Factors

**The following factors must be considered when  
Selecting a software project team structure ...**

1. The **difficulty** of the problem to be solved
2. The **size** of the resultant program(s) in lines of code or function points
3. The **time** that the team will stay together (team lifetime)
4. The **degree** to which the problem can be modularized
5. The required **quality and reliability** of the system to be built
6. The **rigidity**[严格性] of the delivery date
7. The **degree of sociability** (communication) required for the project



# Organizational Paradigms

- **Closed paradigm**—structures a team along a traditional hierarchy of authority
- **Random paradigm**—structures a team loosely and depends on individual initiative [主动性] of the team members
- **Open paradigm**—attempts to structure a team in a manner that achieves some of the controls associated with the closed paradigm but also much of the innovation that occurs when using the random paradigm
- **Synchronous paradigm**—relies on the natural compartmentalization of a problem and organizes team members to work on pieces of the problem with little active communication among themselves

suggested by Constantine [CON93]





# Team Coordination & Communication

- **Formal, impersonal approaches** include software engineering documents and work products (including source code), technical memos, project milestones, schedules, and project control tools (Chapter 23), change requests and related documentation, error tracking reports, and repository data (see Chapter 26).
- **Formal, interpersonal procedures** focus on quality assurance activities (Chapter 25) applied to software engineering work products. These include status review meetings and design and code inspections.
- **Informal, interpersonal procedures** include group meetings for information dissemination and problem solving and “collocation of requirements and development staff.”
- **Electronic communication** encompasses electronic mail, electronic bulletin boards, and by extension, video-based conferencing systems.
- **Interpersonal networking** includes informal discussions with team members and those outside the project who may have experience or insight that can assist team members.



## 2 Product

---

- Software scope
- Problem decomposition



# Software Scope

- **Software Project Scope:**

- **Context.** How does the software to be built fit into a larger system, product, or business context and what constraints are imposed as a result of the context?
- **Information objectives.** What customer-visible data objects (Chapter 8) are produced as output from the software? What data objects are required for input?
- **Function and performance.** What function does the software perform to transform input data into output? Are any special performance characteristics to be addressed?

- Software project scope must be unambiguous and understandable at the management and technical levels.





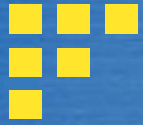
# Problem Decomposition

- Sometimes called **partitioning or problem elaboration**
- Once scope is defined ...
  - It is decomposed into constituent functions
  - It is decomposed into user-visible data objects**or**
  - It is decomposed into a set of problem classes
- **Decomposition process continues until** all functions or problem classes have been defined



## 3 The Process

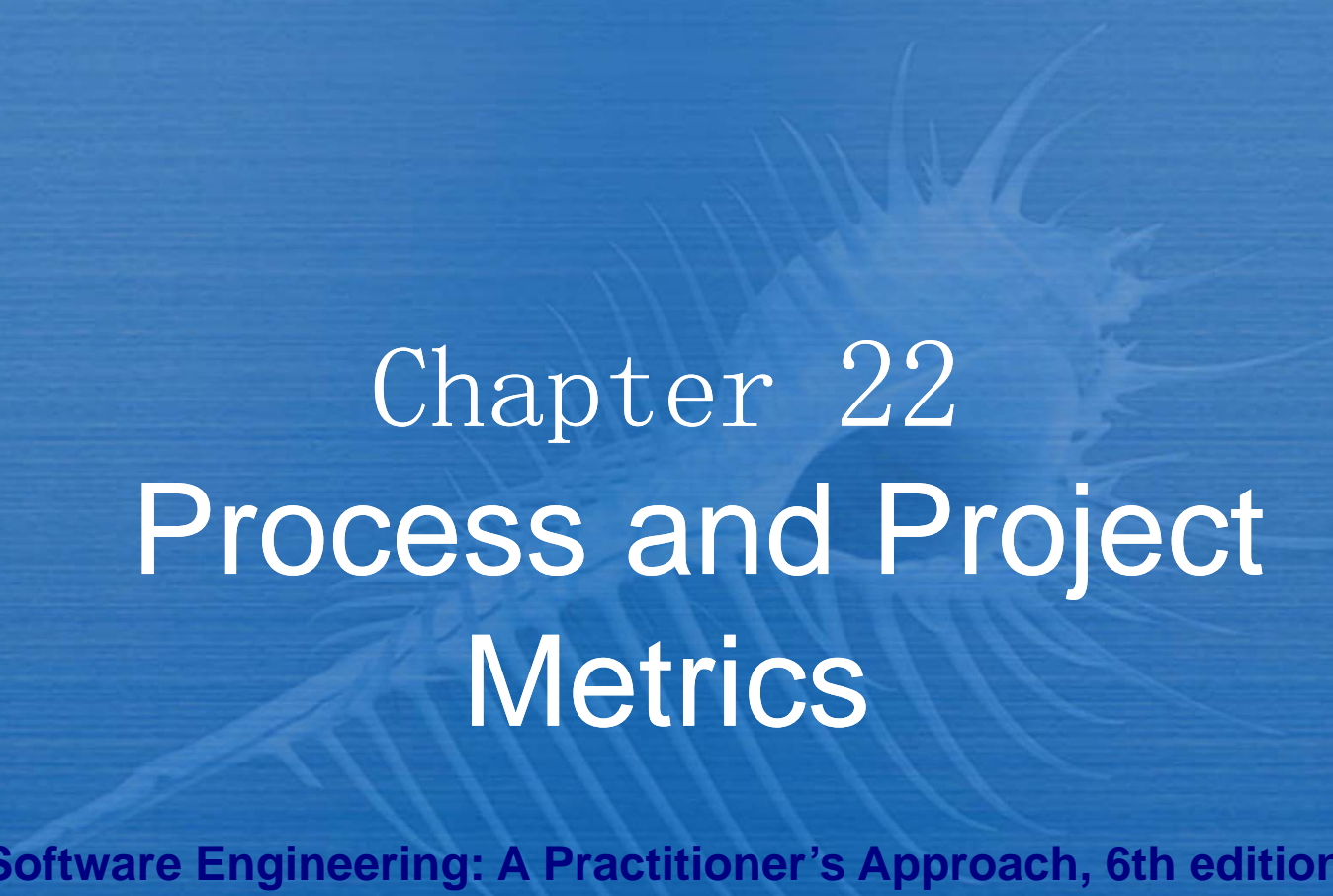
- Once a process framework has been established
  - Consider project characteristics
  - Determine the degree of rigor required
  - Define a task set for each software engineering activity
    - Task set =
      - Software engineering tasks
      - Work products
      - Quality assurance points
      - Milestones[重要事件]



## 4 The Project

- Projects get into trouble[陷入困境] when ...
  - Software people **don't understand** their customer's needs.
  - The product scope **is poorly defined**.
  - Changes are **managed poorly**.
  - The chosen **technology changes**.
  - Business **needs change** [or are ill-defined].
  - Deadlines are **unrealistic**.
  - Users are **resistant**[抵制].
  - Sponsorship is **lost** [or was never properly obtained].
  - The project team **lacks people** with appropriate skills.
  - Managers [and practitioners] **avoid best practices and lessons learned**.

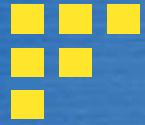




# Chapter 22

## Process and Project Metrics

**Software Engineering: A Practitioner's Approach, 6th edition**  
*by Roger S. Pressman*



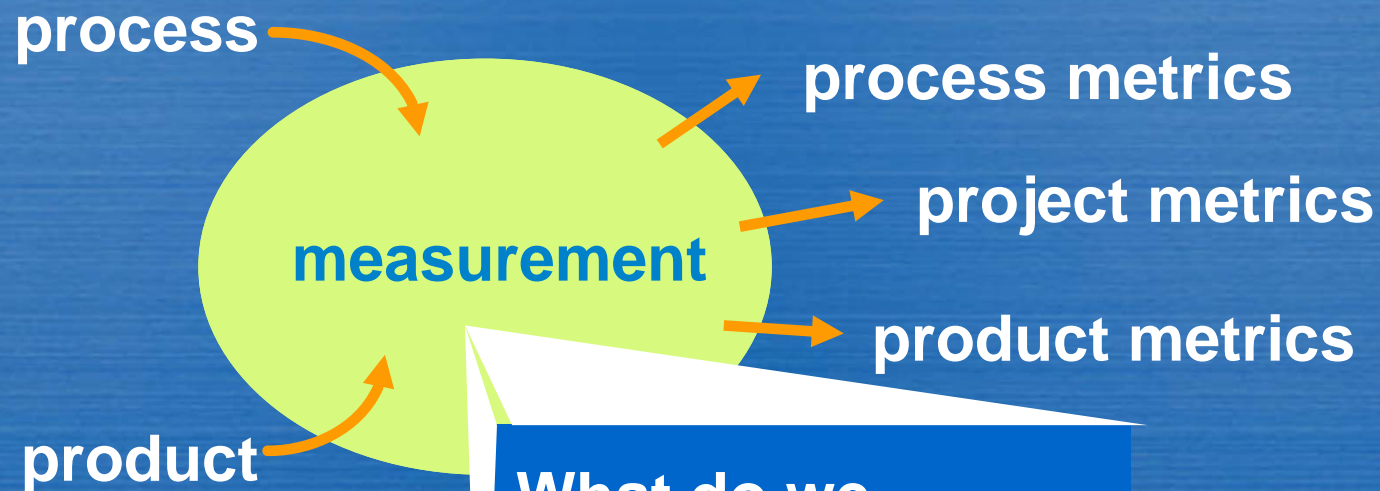
# 本章要点

---

- 过程度量
- 项目度量



# A Good Manager Measures



What do we  
use as a  
basis?

- size?
- function?





# Why Do We Measure?

---

- **assess** the status of an ongoing project
- **track** potential risks
- **uncover** problem areas before they go “critical”
- **adjust** work flow or tasks
- **evaluate** the project team’s ability to control quality of software work products.

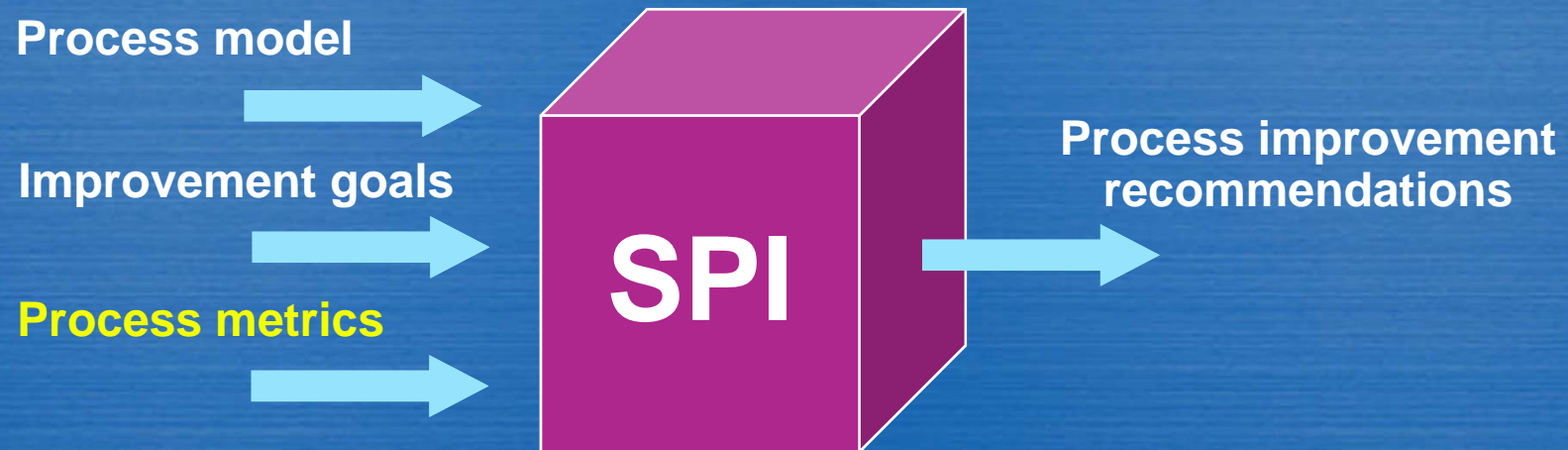


# Process Measurement

- We measure the efficacy[功效] of a software process indirectly. That is, we derive a set of metrics based on the outcomes that can be derived from the process. Outcomes include
  1. measures of **errors** uncovered before release of the software
  2. **defects** delivered to and reported by end-users
  3. work products delivered (**productivity**)
  4. human **effort** expended
  5. calendar **time** expended
  6. **schedule** conformance
  7. **other** measures.
- We also derive process metrics by measuring the characteristics of specific software engineering tasks.



# Software Process Improvement







# Project Metrics

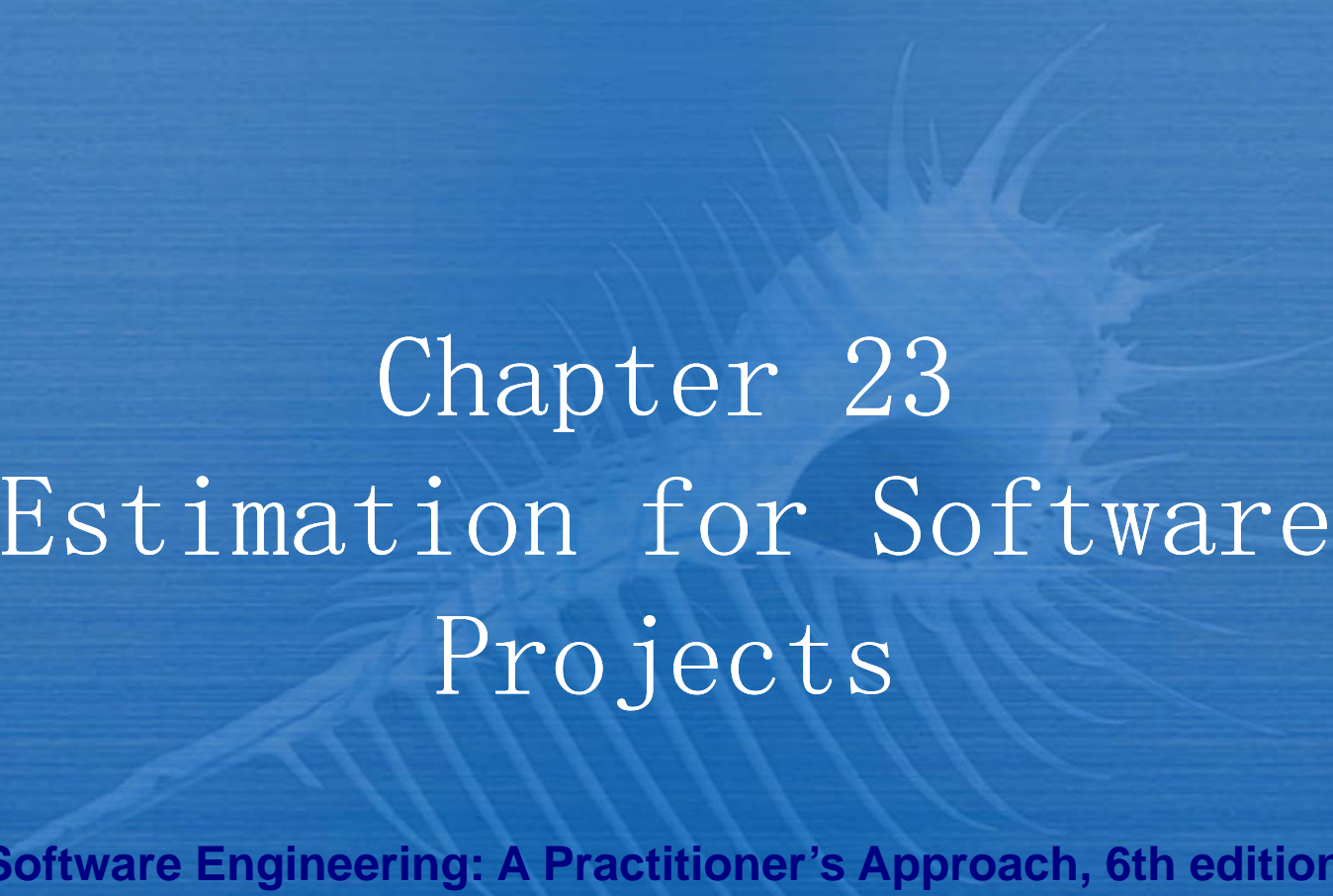
- used to **minimize the development schedule** by making the adjustments necessary to avoid delays and mitigate potential problems and risks
- used to **assess product quality** on an ongoing basis and, when necessary, modify the technical approach to improve quality.
- every project should measure:
  - **inputs**—measures of the resources (e.g., people, tools) required to do the work.
  - **outputs**—measures of the deliverables or work products created during the software engineering process.
  - **results**—measures that indicate the effectiveness of the deliverables.



# Typical Project Metrics

---

- Effort/time per software engineering task
- Errors uncovered per review hour
- Scheduled vs. actual milestone dates
- Changes (number) and their characteristics
- Distribution of effort on software engineering tasks



# Chapter 23

## Estimation for Software Projects

**Software Engineering: A Practitioner's Approach, 6th edition**  
*by Roger S. Pressman*





# 本章要点

## ■ 项目估算

### 项目的哪些属性可以进行估算

软件项目的属性有很多，建议至少以下属性要在项目计划时对其进行估算：

- 1、项目规模
- 2、项目工作量
- 3、项目所需资源
- 4、项目各阶段工作量
- 5、项目成本



# Software Project Planning

The overall goal of project planning is to establish a **pragmatic strategy**[务实的策略] for controlling, tracking, and monitoring a complex technical project.

Why?

So the end result gets done on time, with quality!



# Project Planning Task Set-I

---

- Establish project scope
- Determine feasibility
- Analyze risks
- Define required resources
  - Determine require human resources
  - Define reusable software resources
  - Identify environmental resources





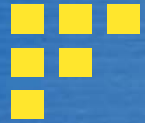
# Project Planning Task Set-II

- **Estimate cost and effort**

- Decompose the problem
- Develop two or more estimates using **size, function points, process tasks** or **use-cases**
- Reconcile the estimates

- **Develop a project schedule**

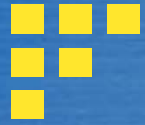
- Establish a meaningful task set
- Define a task network
- Use scheduling tools to develop a timeline chart
- Define schedule tracking mechanisms



# Project Estimation Basis

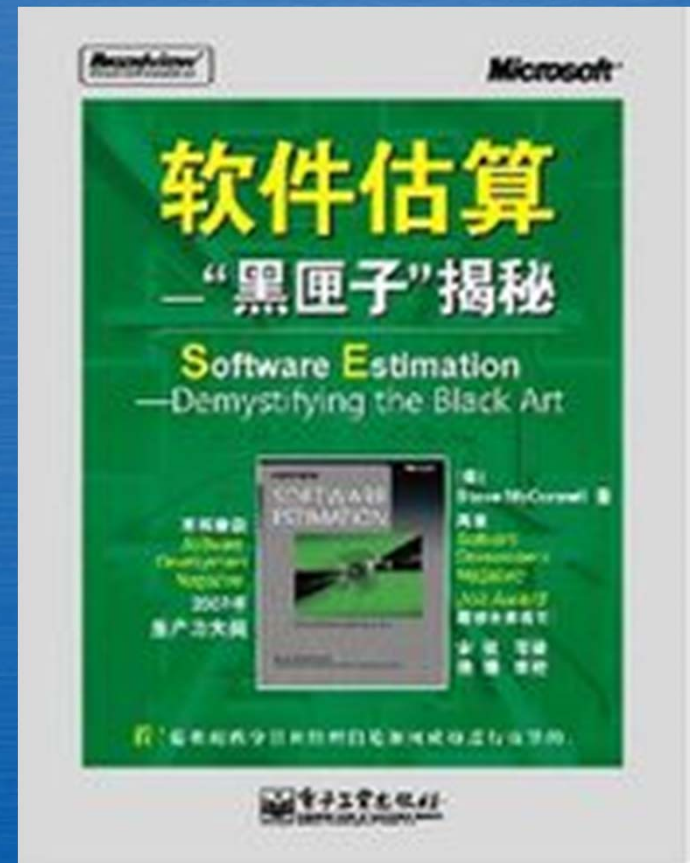


- Project scope must be understood
- Elaboration (decomposition) is necessary
- Historical metrics are very helpful
- At least two different techniques should be used
- Uncertainty is inherent[固有的] in the process



# Estimation Techniques

- 主观和客观方法
- Past (similar) project experience
- Conventional estimation techniques
  - task breakdown and effort estimates
  - size (e.g., FP) estimates
- Empirical models
- Automated tools

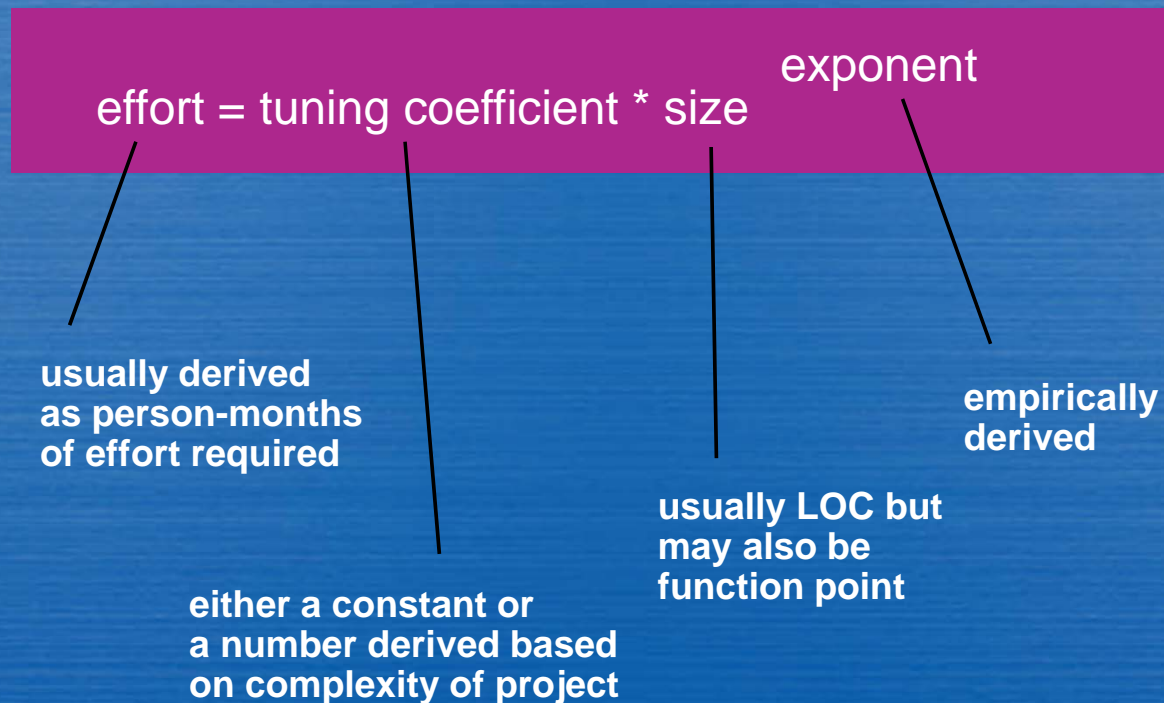






# Empirical Estimation Models

General form:



# COCOMO-II

- COCOMO, 英文全称为constructive cost model, 中文为构造性成本模型。它是一种精确、易于使用的, 基于模型的成本估算方法, 最早由勃姆 (Boehm) 于 1981 年提出。从本质上说是一种参数化的项目估算方法, 参数建模是把项目的某些特征作为参数, 通过建立一个数字模型预测项目成本(类似于居住面积作为参数计算的整体的住房成本)。

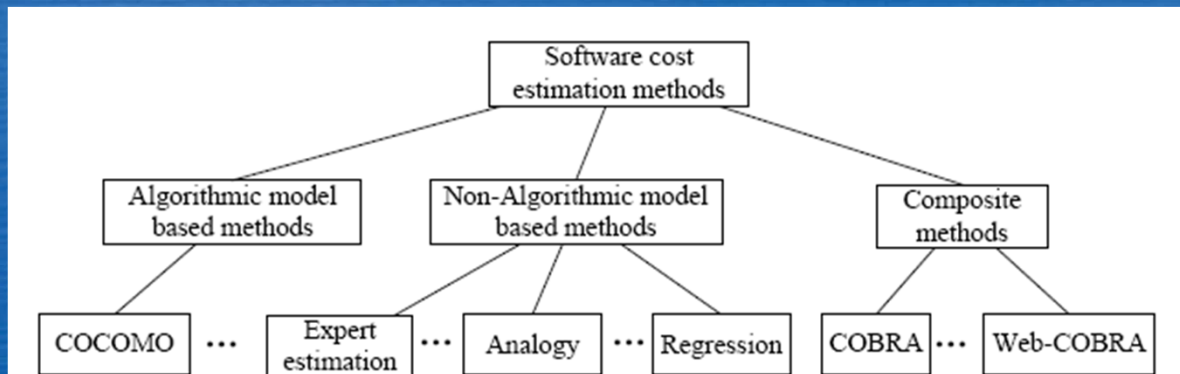


Fig.2 Classification of software cost estimation methods

图 2 软件成本估算方法的分类





# The Software Equation

A dynamic multivariable model

$$E = [\text{LOC} \times B^{0.333}/P]^3 \times (1/t^4)$$

where

**E = effort in person-months or person-years**

t = project duration in months or years

B = “special skills factor”

P = “productivity parameter”





# Estimation for OO Projects

- Develop estimates using **effort decomposition**, **FP analysis**, and **any other method** that is applicable for conventional applications.
- **Using object-oriented analysis modeling** (Chapter 8), develop use-cases and determine a count.
- From the analysis model, determine **the number of key classes** (called analysis classes in Chapter 8).
- Categorize **the type of interface** for the application and develop a multiplier for support classes:

Interface type	Multiplier
• No GUI	2.0
• Text-based user interface	2.25
• GUI	2.5
• Complex GUI	3.0



# Estimation for Agile Projects

- Each user scenario (a **mini-use-case**) is considered separately for estimation purposes.
- The scenario is decomposed into the set of software engineering tasks that will be required to develop it.
- **Each task is estimated separately.** Note: estimation can be based on historical data, an empirical model, or “experience.”
  - Alternatively, the ‘volume’ of the scenario can be estimated in LOC, FP or some other volume-oriented measure (e.g., use-case count).
- Estimates for each task are summed to create an estimate for the scenario.
  - Alternatively, the volume estimate for the scenario is translated into effort using historical data.
- The **effort estimates** for all scenarios that are to be implemented for a given software increment are summed to develop the effort estimate for the increment.