## 1. 算法模型

### COCOMO模型：

创建于1981年，Boehm提出的经典方法，用于根据代码行数（KLOC）估算工作量。

**公式：**Effort = a \* (KLOC)^b，其中a、b是经验参数，Effort为人月。

可分为基本、中级、详细COCOMO。

**优点：**提供了结构化的估算方法。

**缺点：**需要历史数据，无法适应快速变动的需求。

### 改进版COCOMO II：

更适合现代项目（如面向对象、分布式系统）。

考虑到了规模的非线性增长和风险调整因子。

## 2. 专家判断

Delphi法：

由专家独立估算工作量，进行多轮匿名讨论，最终收敛到一致意见。

**优点：**依赖专家知识，适合缺乏历史数据的项目。

**应用场景：**新技术的采用、复杂环境中的估算。

### Wideband Delphi流程

明确任务。

初步估算。

多轮讨论并修订，直到达到共识。

## 3. 类比法（Case-Based Reasoning）

使用历史项目数据，选择与当前项目特征最接近的案例。

公式：欧几里得距离 = √[(目标参数1-历史参数1)² + …]

如：当前项目有7个输入、15个输出，历史项目有8个输入、17个输出，距离=2.24。

**优点：**基于实际经验。

**缺点：**需要完整的历史数据，且适用性依赖案例的相似性。

## 4. 帕金森法

**原则：**工作会消耗所有可用的时间和资源。

**优点：**快速给出粗略的估计。

**缺点：**不适合需要精准控制时间和成本的项目。

## 5. 赢的价格法

**定义：**工作量由市场价格倒推得出。

**应用场景：**预算受限的项目（如政府投标）。

**缺点：**容易低估实际工作量，导致项目失败风险。

## 6. 自顶向下估计

从整体系统需求入手，通过宏观划分估算。

**公式：**Effort = 系统规模 \* 生产率（如每KLOC耗时40天）。

**优点：**适合项目初期规划。

**缺点：**可能忽略细节，导致低估工作量。

## 7. 由低向上估计

**过程：**

将项目分解为具体任务（生成工作分解结构WBS）。

每个任务独立估算，再汇总为总体工作量。

**适用场景：**项目规划后期，有详细设计文档。

**优点：**较为精确。

**缺点：**工作量大，依赖详细信息。

## 8. 功能点法（FP Methods）

**核心概念：**通过功能规模（而非代码量）来评估工作量。

**基本要素：**

外部输入数（EI）、外部输出数（EO）、外部查询数（EQ）、内部逻辑文件数（ILF）、外部接口文件数（EIF）。

**计算公式：**

CT = Σ(信息量 × 权重因子)。

调整：根据复杂性因子。

转换为代码行数（如1功能点=15代码行）。

**适用场景：**

IFPUG方法：适合信息系统。

COSMIC方法：更关注数据移动，适合实时或嵌入式系统。

**优点：**与技术实现无关，标准化程度高。

**缺点：**计算较复杂。