**作业：比较课堂上所讲到的软件工作量估计方法**

# 1. 算法模型

代表方法: COCOMO、COCOMO II。

特点: 使用数学公式（如工作量 = 系数 × 系统规模的指数），结合项目属性进行量化。

优点:

* 定量化，适合大型项目。
* 可通过调整参数适应不同项目类型。

缺点:

* 需要准确的历史数据。
* 参数复杂，需经验丰富的用户。

适用场景: 项目规模大，有较多历史数据支持。

# **2. 专家判断**

代表方法: Delphi法。

特点: 依赖领域专家提供工作量评估，通过多轮匿名讨论达成共识。

优点:

* 灵活性高。
* 对于无历史数据或复杂的新技术项目较为实用。

缺点:

* 主观性较强。
* 结果可能因专家的偏见而有所偏差。

适用场景: 无历史数据或新领域项目。

# **3. 类比估计**

特点: 使用已完成项目与待开发项目的相似性进行比较。

优点:

* 简单直观。
* 不需要复杂的公式计算。

缺点:

* 依赖于与目标项目非常类似的案例。
* 难以量化复杂性。

适用场景: 有类似案例的项目。

# **4. 功能点法（FP法）**

特点: 基于软件功能的五个信息量（输入、输出、查询、内部逻辑文件、外部接口文件）计算工作量。

优点:

* 与实现技术无关。
* 可客观度量不同项目的规模。

缺点:

* 强调数据处理，忽略行为（控制）方面。
* 需要详细的需求定义。

适用场景: 数据驱动的信息系统。

# **5. 参数化模型**

特点: 使用参数化公式计算工作量，考虑多种影响因素。

优点:

* 考虑项目复杂度、团队经验等多种因素。
* 精确度高，适合成熟项目。

缺点:

* 需要大量输入参数。
* 初期项目可能缺乏足够数据。

适用场景: 项目属性明确，资源充分。

# **6. 对象点法（Object Points）**

特点: 计算屏幕、报告和组件等对象的复杂性。

优点:

* 比功能点法更简单。
* 适用于原型开发和面向对象设计。

缺点:

* 对数据驱动系统适应性较差。

适用场景: 面向对象开发和原型设计。

# **7. 分解法**

子类型: 自顶向下估计、由低向上估计。

特点: 将项目划分为子任务，分别评估后汇总。

优点:

* 灵活性高，适合项目规划的不同阶段。
* 可细化到任务级别。

缺点:

* 需对系统结构和任务有清晰理解。

适用场景: 大型项目后期规划。

# **8. 帕金森法**

特点: 基于资源可用性来估算工作量（“工作总是用完所有时间”）。

优点:

* 简单直接。

缺点:

* 可能低估或高估实际需求。

适用场景: 资源有限且必须按时间完成的项目。

# **综合比较**

|  |  |  |  |
| --- | --- | --- | --- |
| 方法 | 优点 | 缺点 | 适用场景 |
| 算法模型 | 定量化、适合大型项目 | 参数复杂、需要历史数据 | 大规模、历史数据丰富 |
| 专家判断 | 灵活性高、适合新领域 | 主观性强 | 无历史数据、新技术项目 |
| 类比估计 | 简单直观 | 依赖类似案例 | 类似项目参考 |
| 功能点法 | 与实现技术无关、客观度量 | 忽略控制功能 | 信息系统 |
| 参数化模型 | 考虑多因素、精确度高 | 数据需求高 | 成熟项目 |
| 对象点法 | 简单易用 | 不适合数据驱动系统 | 面向对象开发 |
| 分解法 | 细化任务 | 对系统理解要求高 | 大型项目规划 |
| 帕金森法 | 简单直接 | 容易高估或低估 | 时间紧迫项目 |

不同方法适用的场景不同，实际选择时需要结合项目规模、复杂性、历史数据和团队经验等因素进行权衡。