第1章：软件工程与软件工程经济学 (期末复习重点)

> **复习提示**：本章主要考察基本概念的记忆和理解，重点关注**软件特点**、**规模度量 (LOC vs FP)** 以及**项目的七大特征**。

1. 软件及其分类与特点

- **软件定义**：程序（指令集合）+ 数据结构 + 相关文档。
- **软件分类**：
  - **按功能**：系统软件、支撑软件、应用软件。
  - **按规模**：微型、小型、中型、大型、甚大型、极大型。
  - **按工作方式**：实时、分时、交互式、批处理。
- **软件特点 (Key Features)**：
  1. **逻辑实体**：具有抽象性，无物理实体。
  2. **无物理磨损**：但在使用中会因变更引入错误而“退化”。
  3. **开发/设计**：不同于传统制造生产。
  4. **成本结构**：主要集中在**开发阶段**（人力成本），无库存成本。
  5. **手工作坊**：尚未完全实现自动化生产。

2. 软件生存周期 (Software Life Cycle)

- **定义**：从概念形成 → 开发 → 使用/维护 → 退役的全过程。
- **典型阶段**：
  1. 系统规划
  2. 需求分析
  3. 设计（概要/详细）
  4. 构建（编码）
  5. 测试
  6. 运行维护

3. 软件开发模型与任务分解

- **开发模型**：
  - **瀑布模型** (Waterfall)：线性顺序，阶段间有反馈。
  - **螺旋模型** (Spiral)：强调风险分析。
  - **增量模型** (Incremental)：分批次交付功能。
- **任务分解 (WBS - Work Breakdown Structure)**：
  - **定义**：将项目分解为可管理的活动层次结构。
  - **分解准则**：按“生存周期阶段”或“目标/功能属性”分解。
  - **作用**：团队组织、进度计划、成本估算的基石。

4. 软件规模与复杂性度量 (⭐ 核心考点)

这是计算题和选择题的高频考点，务必分清 LOC 和 FP 的区别。

A. 规模度量

方法	全称	特点	优缺点
LOC	Lines of Code (源代码行数)	直接测量	**直观**，但**依赖语言** (低级语言行数多，高级语言行数少)。
FP	Function Point (功能点)	间接测量	**与语言无关**，基于系统功能。适合在需求分析阶段估算。

- **FP 计算的关键五大信息量**:
    1. 外部输入 (External Inputs)
    2. 外部输出 (External Outputs)
    3. 外部查询 (External Inquiries)
    4. 外部文件 (External Files)
    5. 内部文件 (Internal Logical Files)

B. 复杂性度量

- 通常利用代码的词汇量（操作符 + 操作数）或控制流结构的复杂度来度量。

5. 软件工程经济学 (SEE)

- **定义**：研究软件工程领域的经济问题。即对技术方案、生产过程、产品服务进行**经济分析、论证、计算和比较。**
- **三大学科支撑**：软件工程技术、软件工程管理学、软件工程经济学。

6. 项目的特点

在既定约束下，为实现特定目的而进行的一次性任务。

- **七大特征**:
    1. 目标性
    2. 相关性
    3. 时限性
    4. 独特性
    5. 约束性
    6. 不确定性
    7. 结果不可逆转性

第2章：软件工程经济学基础 (计算题核心)

> **复习提示**：本章核心在于**“钱”**。重点掌握**资金时间价值的6个公式**（必考计算）以及**现金流量图**的绘制。

1. 软件工程经济分析的三大基本要素

1. **投资与融资**：资金的来源与去向。
2. **成本与效益**：投入与产出的对比。
3. **资源分配**：人力、时间、设备、信息的优化配置。

2. 投资与融资

- **投资**：
  - **生产性投资**：
    - 固定资产（如服务器、房产）：通过**折旧**方式分期计入成本。
    - 流动资产（如原材料、现金）：一次性计入成本，通过销售回收。
    - 无形资产（如专利、商标）：通过**摊销**回收。
  - **非生产性投资**：如购买证券。
- **融资**：
  - **权益性融资**（股票）：无需还本付息，风险共担。
  - **债权性融资**（债券、贷款）：需按期还本付息，风险主要由借款人承担。

3. 成本、收益与利润

- **成本构成**：直接成本（人工、材料）、间接成本、期间费用。
- **利润计算公式**：
  - 销售利润 = 销售收入 - 总成本 - 销售税金及附加
  - 税后利润 = 利润总额 - 所得税
  - *(注意：计算题中常涉及税前与税后利润的转换)*

4. 资金的时间价值 (⭐ ⭐ ⭐ 计算题必考)

- **概念**：资金在生产流通过程中，随时间推移而产生的增值。今天的一块钱 > 明天的一块钱。
- **利息计算**：
  - **单利**：仅本金产生利息。$F = P(1 + n \cdot i)$
  - **复利**：利滚利。$F = P(1 + i)^n$
- **现金流量图 (Cash Flow Diagram)**：
  - **三要素**：大小（资金数额）、流向（箭头上下）、时间点（横轴）。
  - **画图原则**：**首先确定现金流量主体**（是投资方还是承建方？）。箭头向上为流入（收益），向下为流出（投资/支出）。
- **资金等值计算六大公式** (务必熟练背诵与转换):
  - *参数说明*：i (利率), n (期数), P (现值/现在), F (终值/未来), A (年金/等额支付)。

类型	已知	求	公式符号	计算公式
一次支付终值	P	F	$(F/P,i,n)$	$F=P(1+i)^n$
一次支付现值	F	P	$(P/F,i,n)$	$P=F(1+i)^{-n}$
等额支付终值	A	F	$(F/A,i,n)$	$F=A\frac{(1+i)^n-1}{i}$
偿债基金	F	A	$(A/F,i,n)$	$A=F\frac{i}{(1+i)^n-1}$
等额支付现值	A	P	$(P/A,i,n)$	$P=A\frac{(1+i)^n-1}{i(1+i)^n}$
资金回收	P	A	$(A/P,i,n)$	$A=P\frac{i(1+i)^n}{(1+i)^n-1}$

5. 项目评价与决策方法

- **关联矩阵法**：
  - 通过加权平均对多指标方案进行排序。
  - 权重确定常采用**二分比较法**。
- **层次分析法 (AHP)**：将复杂问题分解为目标、准则、方案层，构造判断矩阵，进行一致性检验。
- **模糊综合评价法**：引入隶属度概念，解决评价标准边界不清（如“很好”、“一般”）的问题。

第3章：软件的成本、工期与定价分析

> **复习提示**：本章重点在于**估算方法**的原理区分，以及**价值工程**的分析思路。

1. 软件成本构成

- **开发成本**：人力成本（工资、奖金）、硬件/软件购置费、差旅费、培训费等。
- **维护成本**：改正性、适应性、完善性维护产生的费用。
- **特性**：**人力成本**是软件成本的主要部分（往往占70%以上）。

2. 软件成本与工期的测算方法

- **功能分解法** (Bottom-up)：
  - 将系统拆解为底层功能模块，分别估算后汇总。
  - 利用 β 分布估算期望值：$E=\frac{a+4m+b}{6}$ (a:乐观, m:可能, b:悲观)。
- **Delphi法 (专家判断法)**：
  - 多位专家**背对背**（匿名）估算，经过多轮反馈修正，直至意见趋于一致。避免了权威压制。
- **类比法** (Analogy)：
  - 参考已完成的类似项目，通过调节因子 (AAF) 进行修正。
- **统计模型法 (参数模型)**：
  - **COCOMO模型** (构造性成本模型)：
    - **基本公式**：$MM=a\cdot(KDSI)^b$ （MM:人月, $KDSI$:千行代码）。
    - 分为三种模式：**组织型**（简单）、**半独立型**、**嵌入型**（复杂）。
    - *考点*：嵌入型项目（如实时控制系统）的系数最大，成本最高。

3. 价值工程分析 (Value Engineering, VE)

- **定义**：以最低的寿命周期成本，可靠地实现产品的必要功能。
- **核心公式**：
  $V = \frac{F}{C}$
  - V: 价值 (Value)
  - F: 功能 (Function)
  - C: 成本 (Cost)
- **提高价值的5种途径**：
  1. $F \uparrow, C \downarrow$ (双管齐下，最理想)
  2. $F \uparrow, C \rightarrow$ (成本不变，功能提高)
  3. $F \rightarrow, C \downarrow$ (功能不变，成本降低)
  4. $F \uparrow\uparrow, C \uparrow$ (功能大幅提高，成本略有增加)
  5. $F \downarrow, C \downarrow\downarrow$ (功能略有下降，成本大幅降低)
- **VE对象选择方法**：
  1. **价值系数法**：计算 $VI = FI/CI$。
     - $VI < 1$：成本比重大于功能比重，是**重点改进对象**（需要降低成本）。
  2. **ABC分析法 (Pareto分析)**：
     - 将零件按成本从高到低排列。
     - **A类**：数量少（10-20%），但成本高（70-80%）。**重点管理对象。**
     - B类：中间状态。
     - C类：数量多，成本低。

4. 软件定价

- **成本导向定价**：完全成本加成法（成本+利润+税金）。
- **市场导向定价**：随行就市。

第4章：软件项目的经济效益、社会效益与风险分析

> **复习提示**：本章核心是**"决策"**。不仅要会算 NPV 和回收期，还要知道怎么根据结果选方案（互斥方案排序）。风险部分多为简答题。

1. 单方案项目评价 (核心指标)

- **净现值 (NPV - Net Present Value)**：
  - **定义**：将项目寿命期内各年的净现金流量，按基准收益率 i_0 折现到建设期初的现值之和。
  - **公式**：
    $NPV = \sum_{t=0}^{n} (CI - CO)_t (1 + i_0)^{-t}$
  - **判据**：
    - $NPV \geq 0 \rightarrow$ 方案可行（收益超过了基准回报）。
    - $NPV < 0 \rightarrow$ 方案不可行。

- **内部收益率 (IRR - Internal Rate of Return)**：
  - **定义**：项目在整个寿命期内，$NPV = 0$ 时的折现率。反映了项目**自身的**盈利率。
  - **判据**：$IRR \geq i_0$ (基准收益率) $\rightarrow$ 可行。
- **投资回收期 (N_d)**：
  - **定义**：项目的净收益抵偿全部投资所需要的时间。
  - **静态计算**（不考虑时间价值）：累计净现金流量开始出现正值的年份。
    $N_d = (\text{累计净现金流量出现正值的年份数} - 1) + \frac{\text{上年累计净现金流量的绝对值}}{\text{当年净现金流量}}$
  - **判据**：$N_d \leq N_c$ (基准回收期) $\rightarrow$ 可行。

2. 特殊项目的多方案排序

- **互斥方案**（只能选一个）：
  - **寿命期相同**：直接比较 **NPV**，大的优。
  - **寿命期不同**：不能直接比 NPV。需采用 **净年值法 (NAV)** 或 **最小公倍数法**（将寿命期统一）。
- **收益相同或未知（仅有成本）**：
  - 采用 **费用现值 (PC)** 或 **年费用 (AC)** 最小准则。谁花的钱少选谁。

3. 软件项目的风险分析

- **风险定义**：损失发生的可能性。风险 = 损失 $\times$ 概率。
- **风险识别**：
  - **系统风险**：宏观环境引起，不可控（如政策变化、通货膨胀）。
  - **非系统风险**：微观因素引起，可控（如技术难题、管理不善）。
  - **工具**：风险树、专家调查法。
- **风险分析**：
  - **三点估计法**：对风险变量进行乐观、悲观、可能的估计。
  - **主观概率法**：凭经验判断概率。
- **风险控制**：
  - **策略**：
    1. **回避**（不战）
    2. **转移**（保险、外包）
    3. **分散**（多领域投资）
    4. **承担**（准备风险金）
  - **工具**：因果分析图（鱼骨图）。

第5章：软件生产过程经济分析

> **复习提示**：本章理论性较强，重点理解**Putnam模型**和**N-R曲线**揭示的软件开发规律（尤其是“人月互换”的神话）。

1. 软件生产函数

- **C-D 生产函数 (Cobb-Douglas)**:
  - $Y = AL^{\alpha}K^{\beta}$ (L:劳动, K:资本)。
  - **规模报酬**:
    - $\alpha + \beta > 1$：规模报酬递增（投入增加1倍，产出增加>1倍）。
    - $\alpha + \beta < 1$：规模报酬递减。
- **Putnam 生产函数** (软件专用):
  - $S = C \cdot K^{1/3} \cdot t_d^{4/3}$
  - S: 代码行数 (Size)
  - K: 总工作量 (人年)
  - t_d: 开发时间 (年)
  - C: 技术状态因子
  - **核心结论**：软件开发具有显著的**规模报酬递增**特性。

2. 诺顿-瑞利 (Norden-Rayleigh) 模型

- **N-R 曲线**：描述软件开发过程中人力投入随时间的变化，呈**单峰曲线**（Rayleigh分布）。
  - 人力投入不是均匀的，而是先增后减，在交付期前后达到高峰。
- **开发难度 (D)**:
  - $D = \frac{K}{t_d^2}$
  - 难度与工期的平方成反比。**工期稍微压缩，难度急剧上升。**
- **人力增长率 (D_0)**:
  - $D_0 = \frac{K}{t_d^3}$
  - 想要压缩工期，需要极大的人力增长率支持，往往是不可能的（Brooks法则：向落后的项目增加人手，只会让它更落后）。

第6章：软件项目的进度计划制订与团队组织 (必考)

> **复习提示**：**关键路径法 (CPM)** 是必考的计算/作图题，必须掌握。团队发展的四个阶段常考选择题。

1. 进度计划网络图

- **WBS (工作分解结构)**：进度计划的基础。
- **网络图类型**:
  - **双代号网络图 (AOA)**：箭线表示活动，节点表示事件。
  - **单代号网络图 (AON)**：节点表示活动，箭线表示逻辑关系。
- **逻辑关系**：紧前活动、紧后活动、并行活动。

2. 关键路径法 (CPM - Critical Path Method)

- **目的**：找出决定项目总工期的那条最长路径。
- **时间参数计算** (口诀)：
  - **最早时间 (ES, EF)**：从前往后推，遇到汇聚取**大**值。（必须等所有紧前都做完，我才能开始）。
  - **最晚时间 (LS, LF)**：从后往前推，遇到分支取**小**值。（为了不耽误最后工期，我最晚必须什么时候结束）。
  - **总时差 (TF)**：$TF = LS - ES$ 或 $LF - EF$。活动在不影响总工期前提下的机动时间。
- **关键路径 (Critical Path)**：
  - **定义**：源点到汇点最长的路径。
  - **特征**：关键路径上的活动，**总时差为 0** ($TF = 0$)。
  - **注意**：关键路径可能不止一条；关键活动延误，总工期必延误。

3. 软件项目团队组织

- **团队结构**：
  - **主程序员组** (Chief Programmer Team)：外科手术式，效率高，但过度依赖核心人物。
  - **民主制小组** (Democratic)：人人平等，适合科研/难题攻关，但效率低。
  - **层次式小组** (Hierarchical)：大公司常用，层级管理。
- **团队建设的四个阶段** (Tuckman模型)：
  1. **形成 (Forming)**：相互试探，依赖领导（指导型）。
  2. **震荡 (Storming)**：冲突产生，个性碰撞（影响型）。
  3. **规范 (Norming)**：达成共识，凝聚力形成（参与型）。
  4. **表现 (Performing/辉煌)**：高效协作，聚焦目标（授权型）。

第7章：软件测试的资源分配、进度管理与最优发行(不考)

7.1 软件测试概述

1. 软件测试的目标

- **核心目标**：希望以**最少的人力、费用和时间**，发现软件中潜在的各种差错和缺陷，以期进行改正。
- **必要性**：为此需要建立一定的测试方法、测试策略和测试流程。

2. 软件测试的分类 (重要概念)

分类依据	测试类型	说明
按被测对象	**面向功能/结构**	传统的测试方法。
	面向对象	针对 OO 语言特性的测试。
按是否执行	**静态测试**	**不运行程序。**通过阅读程序查找差错。重点检查：代码与设计的一致性、逻辑正确性、结构合理性、变量定义与使用、循环/分支嵌套、死循环等。
	动态测试	在计算机上执行被测软件。
按内部细节	**黑盒测试**	又称**功能测试**或**数据驱动测试。**不关心内部逻辑，只关心输入输出。
	白盒测试	又称**结构测试**或**逻辑驱动测试。**基于代码内部逻辑结构进行测试。
按系统目标	**功能性测试**	验证功能是否满足需求。
	性能测试	验证响应时间、吞吐量等。
	可靠性测试	验证系统的可靠程度。
	安全性测试	验证系统的安全防护能力。
	强度/恢复测试	验证系统的承受能力和故障恢复能力。

3. 测试阶段划分

- **面向功能/结构的测试阶段**:
    1. **单元测试** (Unit Testing)
    2. **集成测试** (Integration Testing)
    3. **系统测试** (System Testing)
    4. **运行测试** (Operation Testing)
- **面向对象的测试阶段**:
    1. 方法测试

2. 类测试
3. 类簇测试
4. 系统测试
5. 运行测试

4. NIS 软件测试流程

- **基本活动**：
  1. 拟定测试计划 & 编制测试大纲
  2. 设计和生成测试用例
  3. 按序执行：单元测试 → 集成测试 → 系统测试 → 运行测试
  4. 生成测试报告

(注：以下部分为 PPT 目录提及但文本中未详细展开的考点，通常涉及计算和图表分析，请查阅课件补充)

7.2 软件测试的资源分配与进度管理

- **核心问题**：如何在有限的测试时间内（T），分配测试资源（E），以达到预期的可靠性目标。
- **常见模型**：软件可靠性增长模型 (SRGM)，如 **Goel-Okumoto 模型** (指数模型) 或 **S-shaped 模型**。

7.3 软件最优发行问题

- **核心决策**：权衡**测试成本**与**发行后故障带来的维护成本**，确定最佳的软件发行时间（Release Time）。
- **经济学视角**：
  - 测试时间越长 → 测试成本 ↑，发行后维护成本 ↓。
  - 测试时间越短 → 测试成本 ↓，发行后维护成本 ↑。
  - **最优发行点**：总成本（测试成本 + 维护成本 + 延期损失）最低的时间点。