D1

slug	note-1
上次编辑时间	@2026年2月7日 23:15
日期	@2026/02/27
标识	note
状态	进行中

计划

- [x] ~~子解相关应用~~
- [ ] 了解"软件生命周期+开发模型（瀑布/迭代/敏捷）"

重点章节划分

- "软件过程/软件过程模型/软件工程和过程"
- "敏捷开发"
- "需求工程/需求工程过程/需求获取与分析/需求规格"
- "模型建模/UML建模"
- "体系结构设计/软件设计/面向对象设计"
- "软件测试/测试策略/单元测试/集成测试"

💡 软件工程复习主线：

1. 软件工程概述
2. 软件过程（包括瀑布、迭代、敏捷）
3. 需求工程（获取、分析、规格说明）
4. 系统建模（UML：用例、类图等）
5. 软件设计与架构
6. 实现与测试
7. 运行与维护（了解即可）

NOTE

概述

1. 什么是软件工程?
   - 软件工程的核心为用工程化、规范化的方法组织软件开发和维护，强调过程、文档和质量。
   - 软件产品的基本属性是可维护性、可靠性、信息安全性、效率以及可接受性。
   - 软件工程的基本概念为软件过程、可靠性、信息安全性、需求以及复用。

表 1 良性软件的重要属性

产品特性	描述
可接护性	软件必须能够不断进化以满足客户的需求变化。这是软件产品最重要的特性，因为工作环境是不断变化的，软件也必然要跟着变化
可依赖性和安全性	软件可依赖性还包括一些特性：可靠性、保密性、安全性。可靠的软件在系统失败时情况下，也不会造成物理性损害或经济损失。有意图的入侵不能访问或损坏系统
有效性	软件不要浪费内存和处理器等系统资源，因而有效性应包括响应时间、处理时间和资源利用率
可用性	软件必须能被应用，这就意味用户要学、这就意味着，它必须有合理的界面，适用的并且和其他系统是兼容的

2. 软件过程包含的关键活动?

- 需求、设计、实现、测试、维护。
- 软件描述、开发、有效验证、和进化。

三种模型

瀑布模型

迭代模型

敏捷模型

总结

模型对比表

模型	核心思想	优点	缺点	适用场景
瀑布模型	阶段严格按顺序推进，前一步基本定死再下一步。 把整个开发过程分成先需求、再设计、再编码、再测试的线性阶段，基本不回头。	1. 流程清晰，文档完整 2. 易于管理和控制进度 3. 适合大型团队协作	1. 灵活性差，难以应对需求变更 2. 后期发现问题修复成本高 3. 用户要等到最后才能看到成果	需求明确且稳定的项目，如航天、国防等关键系统
迭代模型	把项目拆成多个小版本，每次做一部分功能，不断完善。 早期就能交付部分功能；可以根据每次迭代的反馈调整方向；降低一次性失败的风险。	1. 可以逐步细化需求 2. 早期发现问题，降低风险 3. 每个迭代都有可用版本	1. 需要良好的架构设计 2. 管理复杂度较高 3. 可能出现迭代间的重复工作	需求较清楚但细节待完善的中大型项目
敏捷开发	小步快跑，频繁交付和用户沟通，随需求调	1. 快速响应变化，适应性强 2. 持续交付价	1. 对团队成员要求高 2. 文档相对不完	需求不明确或快速变化的项目，

模型	核心思想	优点	缺点	适用场景
	整。 适用于需求经常变、客户参与度高的小团队互联网项目。	值，用户参与度高3. 团队协作紧密，效率高	善 3. 难以准确预估成本和时间	如互联网产品、创业项目

应用深潜

上机基本流程

1. 看题目 10～20 分钟，理解需求、画图；
2. 设计系统结构（类、模块）；
3. 码代码、调试；
4. 做简单测试；
5. 写一点设计说明/测试报告（如果要求）。

对应“过程模型”里的哪些阶段？

- 题 + 列功能 → 需求分析
- 画用例图、类图 → 设计
- 写代码 → 实现
- 自测 + 写测试用例 → 测试

上机方式更像小瀑布还是小迭代？
上机做题更适合“小迭代”：

- 先把最核心的功能做出来，保证有一个“能跑的版本”；
- 然后再往上加功能和优化结构；
- 避免一开始做太复杂的设计，结果时间不够，连基本功能都做不完。

过程模型：

- 看题 + 列功能 → 需求分析

- 画用例图、类图 → 设计
- 写代码 → 实现
- 自测 + 写测试用例 → 测试