传统模型 (如瀑布、V模型、增量/迭代模型、螺旋模型)：​​

优点：​​

结构清晰，阶段分明：需求、设计、编码、测试等阶段划分明确，易于理解和管理。

文档驱动：产生详尽的文档，方便追溯、审计和交接。

适用于需求明确、变更少的项目：如嵌入式系统、安全关键系统等。

计划性强：早期制定详细计划，预算和进度相对可控（在需求稳定前提下）。

缺点：​​

变更响应慢：后期需求变更代价高昂，可能导致项目返工甚至失败。

客户反馈滞后：客户在项目后期才能看到可运行的软件，无法及时提供反馈。

交付周期长：需要完成所有阶段才能交付最终产品，价值交付延迟。

前期风险可能累积：需求或设计阶段的错误可能在后期才被发现。

灵活性差：难以适应快速变化的市场或需求。

敏捷开发 (主要方法如 Scrum, XP, Kanban)：​​

优点：​​

快速响应变化：通过短周期迭代，拥抱需求变更，灵活适应。

客户/用户持续参与：客户/用户代表深度参与，频繁反馈，确保产品符合预期。

早期并持续交付价值：每个迭代都产出可工作的、潜在可交付的软件增量。

风险早期暴露：每个迭代都进行设计、编码、测试，问题能尽早发现和解决。

提升团队士气和协作：自组织、跨职能团队，共同承担责任，沟通紧密。

聚焦高价值特性：通过优先级排序，确保团队始终在开发最重要的功能。

缺点：​​

对团队要求高：需要高素质、自组织、协作紧密的团队成员。

客户/用户参与度要求高：需要客户/用户代表（如PO）持续投入时间和精力。

前期总体计划模糊：长期精确计划和预算较难，更关注短期目标和适应。

文档相对轻量：更侧重可工作的软件，文档可能不够详尽（需要平衡）。

不适用于所有场景：需求非常明确且固定、法规要求严格文档化、或团队/客户无法满足敏捷要求的项目可能不适用。

### scrum开发

Scrum 是一个帮助团队更好协作的框架。就像橄榄球队（它的名字的由来）为赢得大型比赛进行集训一样，Scrum 鼓励团队从经验中学习，以自组织的方式去处理问题，并对他们的胜利和失败反思，不断改进。虽然 Scrum 通常是被软件开发团队使用，但它的原则和经验可用于各种团队合作，这也是 Scrum 如此受欢迎的原因之一。

Scrum 通常被认为是一个敏捷项目管理框架，它描述了一组帮助协同工作的事件、工件和角色，旨在帮助团队更高效的组织和管理其工作。

三大角色​​

产品负责人​​

职责：最大化产品投资回报，负责管理产品待办列表，对需求排优先级，定义验收标准。

关键实践：

唯一决策者，直接与团队协作。

确保待办列表符合 DEEP原则（粗细适宜、估算过、涌现式、排好优先级）。

开发团队​​

组成：跨职能自组织团队，无固定职称，成员需具备多重技能。

职责：

自主决定每个Sprint的工作量和实现方式。

每日同步进度，协作完成Sprint目标。

ScrumMaster​​

职责：服务型领导，移除团队障碍，确保Scrum流程正确实施，指导组织变革。

关键点：

非项目经理，不分配任务。

与产品负责人角色分离，避免冲突。

关键工件​​

产品待办列表​

动态需求池，包含功能、技术改进、缺陷修复等。

维护原则：

高优先级项细化，低优先级项粗粒度。

每个Sprint梳理，拆分大需求。

Sprint待办列表​​

当前Sprint的任务清单，由团队在计划会议中分解生成。

跟踪工具：任务板和燃尽图。

增量​

每个Sprint结束时交付“完成”的潜在可交付产品增量。

目标：每个Sprint均可发布。

### deepseek_mermaid_20250606_fefaa2

### 关键事件

#### Sprint计划会议：

Part 1：产品负责人讲解高优先级事项，团队确认Sprint目标。

Part 2：团队分解任务，按容量规划承诺工作量。

#### 每日Scrum会议：

三问题同步：

昨日完成什么？

今日计划做什么？

遇到什么障碍？

#### Sprint评审会议：

演示真实软件，利益相关人直接操作。

输出：产品待办列表更新。

#### Sprint回顾会议：

聚焦流程改进：

继续：团队高效实践

改进：障碍根因

避免负面化：需平衡讨论优势与问题。

